SIMULATED MULTI-ECHELON READINESS-BASED INVENTORY LEVELING WITH LATERAL RESUPPLY

THESIS

Todd C. Burnworth, HQ AFMC/A9

AFIT/GOR/ENS/08M-23

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GOR/ENS/08M-23

SIMULATED MULTI-ECHELON READINESS-BASED INVENTORY LEVELING WITH LATERAL RESUPPLY

THESIS

Presented to the Faculty

Department of Operations Research

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Todd C. Burnworth, BS

HQ AFMC/A9

March 2008

AFIT/GOR/ENS/08M-23

SIMULATED MULTI-ECHELON READINESS-BASED INVENTORY LEVELING WITH LATERAL RESUPPLY

Todd C. Burnworth, BS

HQ AFMC/A9

Approved:

_____	_____
Dr. John O. Miller (Chairman)	Date
_____	_____
Dr. Martha C. Cooper (Member)	Date

Abstract

For the past fifty years, U.S. Air Force reparable inventory has been allocated based on an analytic model developed by Dr. Craig C. Sherbrooke. Although versions of his model can be implemented easily with the help of a computer, the analytic approach fundamentally lacks the flexibility to address numerous logistics issues. This body of research will offer a novel alternative approach that will enable researchers to investigate currently unsolved logistics problems such as quantifying the benefits of lateral resupply.

Acknowledgments

I once had a professor who repeatedly reminded his students that *brevity is the soul of wit* (Irani, 2004). I agree, so I will keep this short and hopefully meaningful.

I want to thank my advisor, Dr. John O. Miller. His flexibity, insight and adventurous spirit have been appreciated throughout this endeavor. Additionally, I thank my reader, Dr. Martha C. Cooper, for her extraordinary attention to detail.

I thank Mr. Richard Moore and Mike Niklas of HQ AFCM/A9A for encouraging me to pursue graduate education and being supportive of me while away from the office.

I owe thanks to Mr. F. Michael Slay of LMI for being a continuous source of inspiration, knowledge and friendship.

I owe gratitude to my wife. Her patience, understanding and unconditional love have been tremendously valuable to me.

I need to thank my daughter. In seven months of life, she has already given a lot. I hope to return that favor for the rest of our lives.

Table of Contents

Page

Abstract iv

Acknowledgments v

Table of Contents vi

List of Figures viii

List of Tables x

Introduction 1

- Chapter Overview 1
- Background 2
  - The Reparable Asset Pipeline 2
- Multi-Echelon Models 4
  - Expected Backorders 6
  - Aircraft Availability 6
  - Marginal Analysis 7
  - Readiness-Based Sparing 8
  - Palm's Theorem 8
  - Problem Statement 9
- Research Questions 10
  - Investigative Questions 10
- Scope and Limitations 10

Literature Review 11

- Chapter Overview 11
- Setting Stock Levels 11
  - METRIC 11
  - Multi-Indenture Theory 14
  - VARI-METRIC Multi-Echelon Theory 15
  - DRIVE 22
- Cannibalization 23
- Lateral Resupply 24
- Simulation-Based Leveling 26

Methodology 28

Chapter Overview 28
Phase 1 – Analytic RBL Model 29
Phase 2 – Trace-Driven Simulated Backorders 30
Phase 3 – Simulated Readiness-Based Leveling 36
Modeling Assumptions 38

Results 39

Chapter Overview 39
Phase 1 – Analytic RBL Model 39
Phase 2 – Trace-Driven Simulated Backorders 42
Phase 3 – Simulated Readiness-Based Leveling 50
Lateral Resupply 53

Conclusion 64

Chapter Overview 64
Summary 64
Further Research 65
Future Investigative Questions 65

Appendix: MATLAB Code 67

List of Symbols, Abbreviations and Acronyms 124

Bibliography 127

List of Figures

Page

Figure 1. The Reparable Asset Pipeline (Adapted from Miller, 1995: 16)..... 3

Figure 2. Multi-Echelon Model with CIRF and Lateral Resupply..... 5

Figure 3. Model Verification Concept..... 28

Figure 4. Percent of 1000 Replications Receiving *X* Events..... 32

Figure 5. Comparison of Event-Base Assignment Rules..... 33

Figure 6. Trace-Driven Simulation Flow Diagram..... 35

Figure 7. Image of Analytic Model Example Data (Sherbrooke, 2004, 49-53)..... 40

Figure 8. MATLAB Output for Analytic Model Example..... 41

Figure 9. Observed Backorders vs. Simulated Time Horizon..... 42

Figure 10. Analytic Model Output for Steady State Determination..... 43

Figure 11. Average Expected Backorders for Base Repair Only..... 44

Figure 12. Average Expected Backorders for Depot Repair Only..... 45

Figure 13. Average Expected Backorders for Dissimilar Bases..... 46

Figure 14. Simulated Readiness-Based Sparing Model Output..... 52

Figure 15. Percent EBO Reduction vs. Order & Ship Time (Base-Base)..... 56

Figure 16. Percent EBO Reduction vs. Order & Ship Time (Depot-Base)..... 57

Figure 17. Percent EBO Reduction vs. Average Base Repair Time..... 57

Figure 18. Percent EBO Reduction vs. Depot Demand Rate..... 58

Figure 19. Percent EBO Reduction vs. Average Base Demand Rate..... 58

Figure 20. Percent EBO Reduction vs. Average NRTS Rate..... 59

Figure 21. Percent EBO Reduction vs. Interaction of Average Base Demand Rate and Average Base Repair Time 59

Figure 22. Percent EBO Reduction vs. Average Base Repair Time Divided by the Order & Ship Time (base-base) 60

List of Tables

Page

Table 1. Annual Demand Rates for Dissimilar Bases 46

Table 2. Factor Labels and Experimental Levels 48

Table 3. Factor Labels and Experimental Levels 48

Table 4. Regression Coefficient Statistics for the 2^4 Experimental Design 49

Table 5. Regression Summary Statistics for the 2^4 Experimental Design 49

Table 6. Lateral Resupply Factor Screening Experiment Ranges 55

Table 7. Two-Sample T-Test for Parts Affected by Lateral Resupply 56

Table 8. Specific Data Characterizing Parts A-D 61

Table 9. Lateral Resupply Alternate Factor Ranges 62

Table 10. Two-Sample T-Test for Experiments of Initial and Alt. Factor Ranges 62

SIMULATED MULTI-ECHELON READINESS-BASED INVENTORY LEVELING WITH LATERAL RESUPPLY

Introduction

Chapter Overview

At any given time, the United States Air Force (USAF) accounts for the use of over 100,000 types of reparable spare parts. Each of these parts requires a forecast to drive the provisioning process to maintain a level of stock to buffer against future demands.

In 2007, the Government Accountability Office (GAO) reported that the Air Force's on-hand inventory averaged $31.4 billion during 2002-2005 (Defense Inventory, 2007: 9). During the same years, on average, 64.6% of on-hand inventory was not used to support requirements (Defense Inventory, 2007: 13). The USAF spends approximately six billion dollars on replenishment, annually.

These statistics demonstrate two key concepts. First, the USAF spends a lot of money on inventory. Second, there appears to be a great deal of potential for improvement.

The primary function of this chapter is to provide general background material useful for understanding the problem statement and research objectives.

Background

The Reparable Asset Pipeline

The number of parts required in an inventory system is referred to as a level; this differs with serviceable stock because some stock may be undergoing repair or undergoing shipment. The stock being repaired or shipped contributes to fulfilling the level, but is not immediately available for weapon system application.

As failed parts are removed from weapon systems (e.g. aircraft), they may be repaired onsite or shipped to another location with greater repair capability. Similarly, serviceable stock may be stored onsite or requested from a larger distribution center. This is known as a multi-echelon supply model. The Air Force's repair/supply system has historically been modeled as a dual-echelon model consisting of bases and depots. The base repair is often referred to as an intermediate maintenance location.

As shown in Figure 1, when an aircraft's reparable components fail they are routed from the flight line to a base for triage, the determination of the ultimate repair source. If the base makes the repair, the newly serviceable part is sent to base supply for storage or applied to an existing backorder.

In general, if the base cannot make the repair (determined at triage, or after a failed repair attempt at the base), the failed component is sent to a depot where the part will be repaired or condemned. If the repair is successful, the serviceable asset is stocked in depot supply to be warehoused or applied to an existing depot backorder.

The NRTS% (not reparable this station percentage) or *NRTS rate* is the likelihood that the repair must be outsourced to the depot. Similarly, the *RTS rate* is the probability that the part is repaired at the base. This describes the maintenance portion of the reparable asset pipeline.

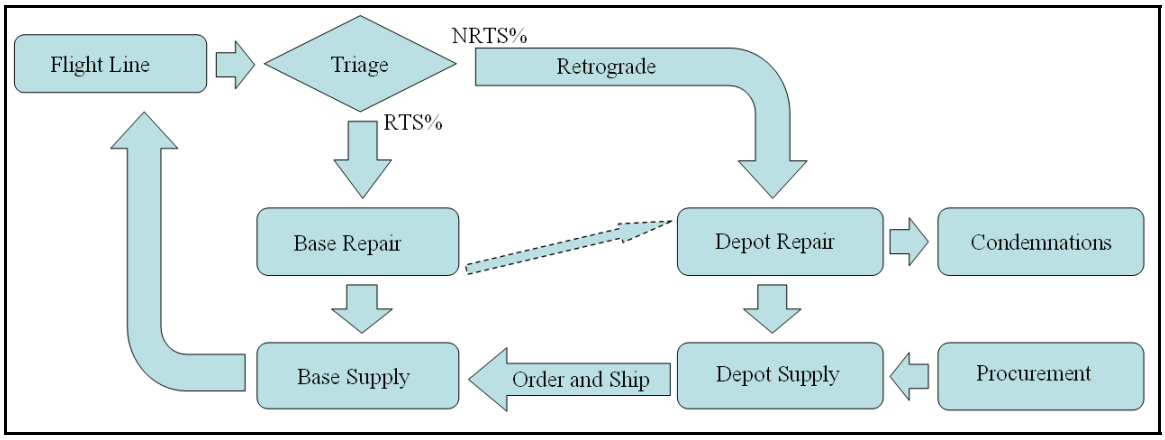

Figure 1. The Reparable Asset Pipeline (Adapted from Miller, 1995: 16)

The supply portion of the reparable asset pipeline begins with the broken component being turned in at the base. At turn-in, the base will attempt to issue a serviceable part. If the broken part must be sent to the depot for repair, the base will request a serviceable part in exchange that can be stocked or applied to the weapon system after the order and ship time (OST) delay.

During this process, the depot will inevitably encounter parts that are damaged or used to the point where a repair is not feasible. The depot will choose to condemn these parts, thus reducing the quantity of parts in the pipeline. Generally, new parts are requisitioned from external supply sources to maintain prescribed stock levels.

A base backorder occurs when a base cannot issue a serviceable part to the flight line at turn-in. A depot backorder occurs when the depot is unable to issue a serviceable

part to a base. A backorder is filled when maintenance fixes a part and it undergoes the appropriate transit time, or when supply is able to locate and procure the asset from another location – if a base procures the asset from another base, this is termed *lateral resupply*. An aircraft waiting for serviceable parts is maintained in NMCS (not mission capable – supply) status; this decreases aircraft availability (AA).

A depot backorder simply means that a part is owed to a base; this does not have any implication about aircraft status. Base backorders are more severe in terms of aircraft availability because the flight line is in need of a part (Abell, 1993: 6).

The pipeline concept is sometimes described by:

$$\mu = \lambda t \tag{1.1}$$

where μ is a part's pipeline quantity, λ is the mean demand rate and t is the mean resupply time. Further, if we let r be the RTS rate, BRT be the base repair time, and DDT be the depot delay time (DDT accounts for retrograde time, the time it takes to ship from the base to the depot); this equation can be modified (Miller, 1995: 22):

$$\mu = \lambda\left[(r)BRT + (1-r)(OST + DDT)\right] \tag{1.2}$$

Multi-Echelon Models

As discussed previously, the Air Force has historically used a dual-echelon supply model. If more stock is located at the bases, the aircraft requiring the spares will be serviced quickly. However, if the stock is misallocated, a base requiring the parts may have trouble locating a serviceable spare. If lateral resupply is assumed to be non-existent, it would be desirable to hold stock at the depot to be shipped (requiring an OST

delay) to the proper location. If too much stock is held at the depot, the OST delay would regularly impact aircraft availability. Stock allocation tradeoffs are necessary in this scenario; some stock should be held at a centralized location for ease of distribution, and some should be held at the point of use.

In recent years, Centralized Intermediate Repair Facilities (CIRFs) have been implemented to act as a common repair organization for specific types of parts for a number of bases. This system is outlined in figure 2.

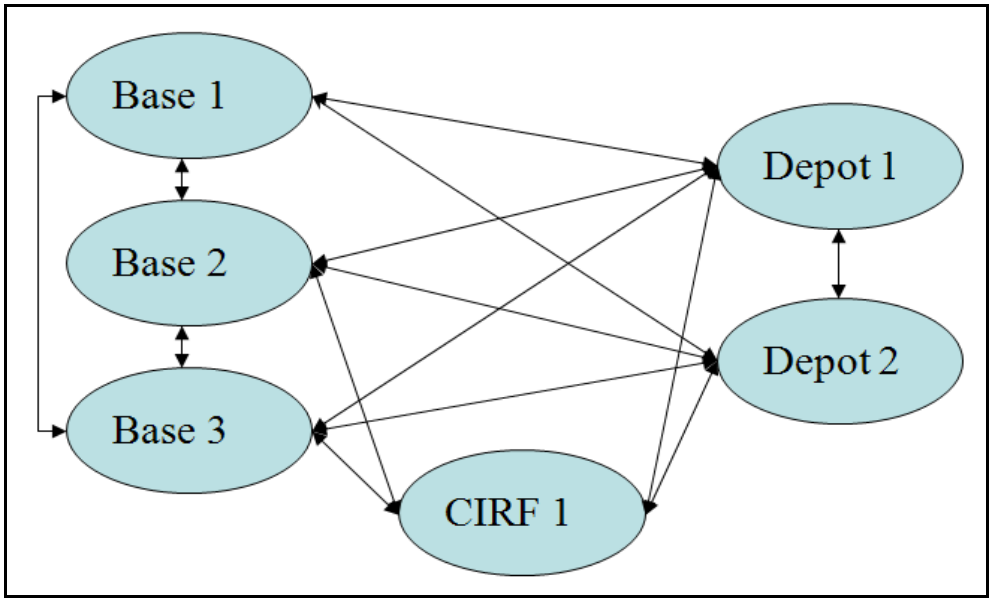

Figure 2. Multi-Echelon Model with CIRF and Lateral Resupply

In this scenario an additional transportation delay may be incurred in the pipeline process due to the CIRF. This has the effect of decreasing availability over the Air Force as a whole. Higher stock levels can make up for the loss in availability, but can be costly. The rationale is that centralizing repair activities saves more money than what is lost due to maintaining availability targets with the additional inventory and transportation delays.

Expected Backorders

A backorder (BO_i) can be explicitly defined as the inventory condition that a serviceable part *i* is requested for which there is no stock. The expected backorder ($EBO_i(s_i)$) is the expected number of outstanding backorders at a random point in time for a given stock level *s* for part *i*. To elicit the probabilities used in the $EBO_i(s_i)$ calculation, distributional assumptions are made regarding each part's pipeline (undergoing procurement, repair or in transit) for each location (Miller, 1995:18). There is a direct link between EBO minimization and maximizing availability (O'Malley, 1983: 2-6).

Aircraft Availability

In 1954, the Department of Defense (DoD) began examining ways to standardize the spares provisioning process (Orsburn, 1991: xiii). With the advent of practical computing and digital data storage, the 1960's were the beginning of a new era of logistics-based operations research.

It was during this period that system performance metrics gained popularity. Availability (A) surfaced as the primary performance metric, and is defined as the expected percentage of a fleet of aircraft that is not down for spares at a random point in time (O'Malley, 1983:2-6).

$$A = 100\prod_{i=1}^{l}\left\{\frac{1-EBO_i(s_i)}{NZ_i}\right\}^{Z_i} \qquad (1.3)$$

To measure availability, one must know the specific part types used in the weapon system (l), the fleet size (N), the number of parts per weapon system (Z_i), and the $EBO_i(s_i)$. A key observation is that availability is maximized by minimizing $EBO_i(s_i)$. This concept will be revisited in Chapter 2.

Marginal Analysis

The USAF Materiel Command's overarching goal is to maximize availability through expected backorder minimization; this must be done within budgetary constraints. Consequently, resupply actions are taken based on a technique referred to as marginal analysis. Marginal analysis is a greedy heuristic to determine which part provides the greatest reduction in EBOs per dollar spent. A resupply action (i.e. repair or procure decision) for a single part i with a large sort value (sv) is performed with higher priority than an action with a small sort value where the sort value is defined as:

$$sv_i = \frac{EBO_i\left(s_i - 1\right) - EBO_i(s_i)}{c} \tag{1.4}$$

where c is the cost of the action. This is the mathematical tool used to establish requirements and resupply actions, and is often thought of as getting the most "bang for the buck" in terms of availability.

As of 1973, the Air Force's marginal analysis algorithms relied on a near-Poisson demand process assumption (via EBO calculation) that set the variance-to-mean ratio (VMR) for all recoverable parts at 1.01. R.J. Stevens and J.M. Hill performed an empirical analysis that showed VMRs were typically not close to the 1.01 standard (Stevens, 1973: *i*). Since this study, there have been many proposals for changes in the

distributional assumptions, but the negative binomial distribution (a generalization of the Poisson) is used in the majority of models today.

Readiness-Based Sparing

Readiness-based sparing is the process of calculating stock levels for locations with the objective of minimizing system EBOs. The Air Force accomplishes this through a model called Readiness-Based Levels (RBL). After the Aircraft Availability Model (AAM) computes worldwide requirements, the requirements are passed to RBL to be split into levels for each location.

RBL is largely based on the METRIC (Multi-Echelon Technique for Recoverable Item Control) model developed in 1968 by Craig C. Sherbrooke. An expected backorder at one location is weighted the same as an expected backorder for the same part elsewhere. RBL does not account for the impact of lateral resupply, CIRFs, or shipping times (or costs) between locations. The levels are issued quarterly (Long, 1996: 5).

Palm's Theorem

Palm's theorem can be expressed (Muckstadt, 2005: 43):

> *Suppose demands occur according to a compound Poisson process where λ is the customer order arrival rate. Suppose also that the resupply times are independent and identically distributed with density g(t) with mean τ. Assume when a customer order is received, the resupply time for all units in the order is the same and is drawn from the resupply time distribution. The steady state probability of x units in resupply is given by the compound Poisson distribution with mean $\lambda\tau\mu$, where μ is the average customer order size.*

This result has also been extended to logarithmic Poisson processes, where cluster interarrival times are exponentially distributed and cluster size is logarithmic. Despite the desirable *independent increments* property (future states are not dependent on current

states), logarithmic Poisson processes have a constant mean and variance; empirical data has shown that demand means and variances change with time, and further, that clusters are not observed. The negative binomial distribution maintains *independent increments*, necessitates a variance larger than the mean, and is discrete for non-negative values. For these reasons, it is used to model inventory pipelines (Sherbrooke, 2004: 64). Palm's Theorem is assumed to be applicable.

As a result of Palm's Theorem, the Air Force does not attempt to model the repair or transportation time distributions. Demand rates are exclusively used to model the pipeline distributions that are used for EBO estimation. If the demand distribution and repair distribution are not independent, Palm's Theorem is violated; this can occur with repair capacity constraints or with a small number of aircraft (Cochran, 2002: 530).

As discussed previously, the EBO estimates are used to calculate inventory levels and to make resupply decisions via marginal analysis. It should be noted that the Palm's Theorem assumption is fundamental to building the distributions used in marginal analysis.

Problem Statement

As the Air Force continues to transform its business processes to more efficiently use its budget to provide support to the warfighter, a need exists to provide rapid what-if analysis regarding policy or procedural changes in operations, as well as to test the sensitivity of underlying model assumptions to efficiently guide future research initiatives. The goal of this research project is to provide a flexible model of reparable

asset supply chain processes that can assess a change's impact on base expected backorders (an established proxy for aircraft availability) and optimal stock levels.

Research Questions

Can a tool be built to assess the impact of policy or procedural changes (such as lateral resupply) on reparable assets in terms of warfighter support?

Investigative Questions

1.) Does simulation-based marginal analysis provide opportunities to improve supply chain and weapon system support?

2.) Does modeling base lateral resupply enable more realistic projections of system availability and improved inventory solutions? Previous attempts to answer this question have been unsuccessful (Sherbrooke, 2004: 258).

3.) Can a variance/bias reduction technique be used to improve the model output?

Scope and Limitations

The investigative questions posed above are only a small fraction of the breadth of questions that can be addressed with the appropriate simulation model. The purpose of this project is to make progress towards a new method of inventory leveling that will enable research of many more topics than is possible with legacy models. After this research project concludes, future research topics will be left for other students' undertaking.

Literature Review

Chapter Overview

The first chapter provided background information necessary for understanding the problem statement, objectives and scope of this research. This chapter will serve to provide a detailed summary of previous work relevant to this project's objectives, methodologies and future applications.

The Air Force currently sets stock levels using an approach developed over the past five decades. Researchers have gone to great length to analytically model multi-indenture, multi-echelon systems. Attempts have also been made to analytically describe item cannibalization and lateral resupply. A description of these approaches (with details of limiting assumptions) is provided in this chapter. The reader should note the escalating complexity with respect to modest modeling gains. This chapter will reinforce the need for an alternative way to model EBOs and perform readiness based leveling.

Setting Stock Levels

METRIC

The Multi-Echelon Technique for Recoverable Item Control (METRIC) model was proposed to the Air Force by Craig C. Sherbrooke in 1968 while employed at the RAND Corporation. The model seeks to minimize worldwide expected base backorders

for reparable first-indenture items. These parts are typically expensive and ordered individually.

Sherbrooke outlines five key model assumptions (Sherbrooke, 2004: 46-47):

1.) *The decision as to whether a base repairs an item does not depend on stock levels or workload.*
2.) *The base is resupplied from the depot, not by lateral supply from another base.*
3.) *The (s-1, s) inventory policy is appropriate for every item at every echelon; subsequently administrative and holding costs are not needed for optimization.*
4.) *Optimal steady-state stock levels are determined.*
5.) *Repair cost is always less than purchase cost.*

The first assumption implies that a base will always make a repair when it has the capability, and that any parts needed for the repair will be ordered from the depot and received in a timely fashion. Sherbrooke makes the second assumption because he believes bases are rarely supplied from other bases and it simplifies the transportation cost structure. The (s-1, s) policy is simply one-for-one ordering; when a part breaks, another is requisitioned to increase stock back to the prescribed level. Because the one-for-one policy specifies the order quantity, administrative and holding costs are considered the same for all orders and not needed for optimization. The fourth assumption relates to a belief that the number of aircraft and flying hours remain constant over some time horizon. The final assumption results in favoring repairing broken items before purchasing new ones.

The stock balance equation forms a basis for future discussion, it shows that the stock level (s) is equal to the on-hand inventory (OH) plus the inventory due in (DI) minus any existing backorders, or (Sherbrooke, 2004: 24):

$$s = \text{OH}+\text{DI}-\text{BO} \tag{2.1}$$

where due-ins for a depot are from repair, and for a base can be from resupply (the depot) or base repair.

For a single first-indenture item, Sherbrooke defines the following variables and system of equations; a subscript of zero refers to the depot, whereas a positive subscript *j* refers to a base (Sherbrooke, 2004: 48):

m_j = average annual demand at base *j*
t_j = average repair time (in years) at base *j*
μ_j = average pipeline at base *j*
r_j = probability of repair at base *j*
O_j = average order-and-ship time from depot to base *j*

A depot's average annual demand is the percentage of base demands that could not be repaired at the base, or:

$$m_0 = \sum_{j=1}^{J} m_j (1 - r_j) \tag{2.2}$$

The METRIC model assumes that demands from each base is a Poisson process, thereby allowing the processes to sum into a larger Poisson process. The resulting average depot pipeline is $m_0 t_0$; subsequently, depot expected backorders can be expressed: $EBO(s_0 | m_0 t_0)$. This permits formulation of the average pipeline for a part at base *j* (where $j > 0$):

$$\mu_j = m_j \left(r_j t_j + (1 - r_j) \left\{ O_j + \frac{\mathrm{EBO}(\mathrm{s}_0 \mid \mathrm{m}_0 \mathrm{t}_0)}{\mathrm{m}_0} \right\} \right) \tag{2.3}$$

It is worth noting that because an expected depot backorder is a time-weighted average number of backorders, dividing by average demand yields the average depot delay to obtain a serviceable part to ship. If sufficient depot OH stock is kept, the

expected backorders should be small, and the base must only wait the O_j to receive the part.

To optimize levels, METRIC begins by setting worldwide base and depot stock levels to zero. It increments worldwide base or depot stock levels by a single unit to greedily reduce worldwide base EBOs via marginal analysis. Subsequently, the worldwide base stock must then be subdivided amongst individual bases; this is also accomplished through marginal analysis in the RBL system.

Multi-Indenture Theory

An aircraft part's indenture is its location on a bill of materials in relation to the airframe. For example, a first-indenture part is a large assembly taken directly off the aircraft such as an engine. This part is typically held and repaired at the base, and is subsequently called a line replaceable unit (LRU). On the LRU, there are smaller components that may be taken off that are repaired at the base if possible, but may be easily shipped to the depot for repair. These types of components are called shop replaceable units (SRU) and represent the second level of indenture. A common modeling assumption is that the failure of an LRU is due to the single failure of an SRU (Sherbrooke, 2004: 65).

Using Sherbrooke's notation, a zero subscript for the index i refers to an LRU and $i > 0$ refers to an indentured SRU. If one assumes that the average demand rate for an LRU (m_0) is Poisson, then each of its SRUs must have average demand rates (m_i) that are also Poisson and independent of each other (Sherbrooke, 2004: 65).

$$m_0 = \sum_{i=1}^{I} m_i \tag{2.4}$$

Equation 2.4 is interesting, because in practice a single m_i can exceed the value of m_o. An example of this provided by F. Michael Slay is when a lightbulb fails and can be replaced without generated a demand for the LRU. Sherbrooke assumes this equation sufficiently models reality; others disagree.

Additionally, Sherbrooke defines t_0 as the LRU average repair time and t_i as the SRU average repair time. The expected pipeline for an LRU is (Sherbrooke, 2004: 66):

$$E[X_0] = m_0 t_0 + \sum_{i=1}^{I} EBO(s_i \mid m_i t_i) \tag{2.5}$$

This is just the LRU demand rate multiplied by the LRU repair time plus the expected number of LRUs waiting on SRUs from the depot.

VARI-METRIC Multi-Echelon Theory

The VARI-METRIC model is a multi-echelon, multi-indenture model that was built by Slay in 1984 and was based on the METRIC model. Whereas METRIC focused primarily on pipeline means, VARI-METRIC also used pipeline variances to calculate improved estimates of backorders.

Discussion regarding this topic begins with the definition of conditional variance. The variance for the number of parts in the pipeline at base j is dependent on the number of parts in repair at the depot, x_o (Sherbrooke, 2004: 103-106):

$$Var[X_j] = E[Var(X_j \mid X_0)] + Var[E(X_j \mid X_0)] \tag{2.6}$$

Because base pipelines are partially composed of a time segment dependent on the number of units in depot repair x_0, the expected pipeline for base j can be written as a conditional expectation:

$$E[X_j] = \sum_{x_0=1}^{\infty} E[X_j | X_0] \Pr\{X_0 = x_0\} \tag{2.7}$$

Given a number of units in depot repair C, one can compute the conditional expectation for the number of units in the base pipeline. If $x_0 \leq s_0$ there are no depot backorders due to the depot repair time because depot stock can supply the base over the order and ship time. Therefore, the expected number of units in the base pipeline is the average demand rate multiplied by the order and ship time (assumed to be the same for all bases for reasons to be discussed later).

$$E[X_j \mid x_0] = m_j O \quad \forall \quad x_0 \leq s_0 \tag{2.8}$$

When x_0 exceeds s_0, a finite number of backorders occur; the fraction of backorders allocated to a base is proportional to the base's average demand and is constant because of the use of rates. This allows use of the binomial distribution to allocate the backorders amongst bases, and will be used later.

$$E[X_j \mid x_0] = m_j O + \frac{m_j (x_0 - s_0)}{m_0} \quad \forall \quad x_0 > s_0 \tag{2.9}$$

After taking the expectation with respect to x_0:

$$E[X_j] = m_j O + \frac{m_j EBO(s_0)}{m_0} \quad \forall \quad x_0 > s_0 \tag{2.10}$$

By employing the variance operator with respect to x_0, the first term is equal to zero; this leaves the following:

$$Var\left[E\left(X_j \mid X_0\right)\right] = \frac{m_j^2 VBO\left(s_0\right)}{m_0^2} \quad \forall \quad x_0 > s_0 \tag{2.11}$$

To satisfy the requirements for the conditional variance equation, $Var(\mathrm{X_j} \mid \mathrm{X_0})$ must also be defined. $E(\mathrm{X_j} \mid \mathrm{X_0})$ has previously been determined; using this and that base backorders are binomially distributed amongst bases, applying the variance operator to terms containing x_0 results in:

$$Var\left[X_j \mid x_0\right] = \begin{cases} m_j O & \forall \quad x_0 \le s \\ m_j O + \dfrac{m_j}{m_0}\left(1 - \dfrac{m_j}{m_0}\right)\left(x_0 - s_0\right) & \forall \quad x_0 > s \end{cases} \tag{2.12}$$

After an additional use of the expectation operator, the conditional variance can be shown to be:

$$Var\left[X_j\right] = m_j O + \frac{m_j}{m_0}\left(1 - \frac{m_j}{m_0}\right) EBO\left(s_0\right) + \frac{m_j^2 VBO\left(s_0\right)}{m_0^2} \tag{2.13}$$

To be consistent with prior notation we define the following terms with SRU i (i = 0 represents an LRU) and base j ($j = 0$ refers to the depot) (Sherbrooke, 2004: 106):

m_{ij} = average annual demand for SRU i at base j
t_{ij} = average repair time (in years) for SRU i at base j
r_{ij} = probability of repair for SRU i at base j
q_{ij} = conditional probability that an LRU repaired at base j will result in SRU i being identified as the cause
O_i = order and ship time from depot to any base of SRU i (assuming the depot has stock available and the time is the same for all bases)
s_{ij} = stock level for SRU i at base j
X_{ij} = number of units of SRU i at base j in resupply at any point in time
EBO($s \mid \mu, Var$) = expected backorders for stock level s with pipeline mean μ and variance Var

The average annual demand for SRU i can be shown to be the average annual LRU demand rate multiplied by the probability it is repaired at base j and that the SRU i has been identified as the cause of the failure (Sherbrooke, 2004: 107-111).

$$m_{ij} = m_{0j} r_{0j} q_{ij} \tag{2.14}$$

The depot's average annual LRU demand is the sum of the base LRU average annual demands multiplied by the probability they are outsourced to the depot.

$$m_{00} = \sum_{j=1}^{J} m_{0j} \left(1 - r_{0j}\right) \tag{2.15}$$

The depot's average annual demand for SRU i can be shown similarly, but contains a portion for when an SRU is demanded for a repair of an LRU already at the depot.

$$m_{i0} = \sum_{j=1}^{J} m_{ij} \left(1 - r_{ij}\right) + m_{00} q_{i0} \tag{2.16}$$

Of all the SRU *i* demanded at the depot, the fraction that is due to LRU depot demand is:

$$f_{i0} = \frac{m_{00} q_{i0}}{m_{i0}} \tag{2.17}$$

The expected number of LRUs in depot repair is a combination of the average LRU demand rate at the depot with the associated repair time and the LRUs that are being delayed due to SRU *i* depot backorders.

$$E[X_{00}] = m_{00} t_{00} + \sum_{i=1}^{I} f_{i0} EBO(s_{i0} \mid m_{i0} t_{i0}) \tag{2.18}$$

For a given total number of SRU *i* depot backorders and the fixed probability f_{i0} of a particular backorder delaying an LRU at the depot, the binomial distribution is used to describe the probability of any number of the backorders delaying an LRU. This permits formulation of the variance of the number of LRUs in depot repair.

$$Var[X_{00}] = m_{00} t_{00} + \sum_{i=1}^{I} f_{i0}(1 - f_{i0}) EBO(s_{i0} \mid m_{i0} t_{i0}) + \sum_{i=1}^{I} f_{i0}^2 VBO(s_{i0} \mid m_{i0} t_{i0}) \tag{2.19}$$

Of all the SRU *i* demanded at the depot, the fraction that is needed at base *j* is:

$$f_{ij} = \frac{m_{ij}(1 - r_{ij})}{m_{i0}} \tag{2.20}$$

The expected number of SRU *i* in base repair or resupply is a combination of the average SRU *i* demand rate at base *j* and the time it takes to obtain a serviceable part directly from the depot or for the base to repair it, and the expected delay from the depot for SRU *i* depot backorders.

$$E[X_{ij}] = m_{ij}[(1 - r_{ij}) O_i + r_{ij} t_{ij}] + f_{ij} EBO(s_{i0} \mid m_{i0} t_{i0}) \tag{2.21}$$

For a given total number of SRU *i* depot backorders and the fixed probability f_{ij} of a particular SRU *i* backorder being for base *j*, the binomial distribution is used to describe the probability of any number of the backorders belonging to base *j*. This permits formulation of the variance of the number of SRU *i* in repair or resupply for base *j*.

$$Var[X_{ij}] = m_{ij}[(1-r_{ij})O_i + r_{ij}t_{ij}] + f_{ij}(1-f_{ij})EBO(s_{i0} \mid m_{i0}t_{i0}) + f_{ij}^2 VBO(s_{i0} \mid m_{i0}t_{i0}) \quad (2.22)$$

Of all the LRUs demanded at the depot, the fraction that is due to LRU demand at base *j* is:

$$f_{0j} = \frac{m_{0j}(1-r_{0j})}{m_{00}} \quad (2.23)$$

The expected number of LRUs in base repair or resupply for base *j* is a combination of the average LRU demand rate at base *j* and the time it takes to obtain a serviceable part directly from the depot or for the base to repair it, the expected delay from the depot for LRU depot backorders, and the expected delay for each of the LRU's indentured SRUs' backorders at base *j*.

$$E[X_{0j}] = m_{0j}[(1-r_{0j})O_0 + r_{0j}t_{0j}] + f_{0j}EBO(s_{00} \mid E[X_{00}], Var[X_{00}]) + \sum_{i=1}^{I} EBO(s_{ij} \mid E[X_{ij}], Var[X_{ij}]) \quad (2.24)$$

For a given total number of LRU depot backorders and the fixed probability f_{0j} of a particular LRU backorder being for base j, the binomial distribution is used to describe the probability of any number of the backorders belonging to base j. This permits formulation of the variance of the number of LRUs in repair or resupply for base j.

$$\begin{aligned} Var\left[X_{0j}\right] = m_{0j}\left[\left(1-r_{0j}\right)O_0 + r_{0j}t_{0j}\right] + f_{0j}\left(1-f_{0j}\right)EBO\left(s_{00} \mid E\left[X_{00}\right], Var\left[X_{00}\right]\right) + \\ f_{0j}^2 VBO\left(s_{00} \mid E\left[X_{00}\right], Var\left[X_{00}\right]\right) + \sum_{i=1}^{I} VBO\left(s_{ij} \mid E\left[X_{ij}\right], Var\left[X_{ij}\right]\right) \end{aligned} \tag{2.25}$$

The overarching goal is to allocate stock in such a manner as to maximize aircraft availability. The availability at base j for a particular LRU is (Sherbrooke, 2004: 111):

$$A_{0j} = 100\left\{1 - \frac{EBO\left(s_{0j} \mid E\left[X_{0j}\right], Var\left[X_{0j}\right]\right)}{N_j Z_0}\right\}^{Z_0} \tag{2.26}$$

where N_j is the number of aircraft residing at base j and Z_0 is the number of occurrences of the LRU on the aircraft (quantity per application, or QPA). To find the availability for a group of similar aircraft, the equation must be adjusted to consider all LRUs and the number of aircraft. Marginal analysis is used to make tradeoffs between depot stock and worldwide-base stock, and then to subdivide worldwide-base stock amongst bases.

Modifying this logic is possible, but may lessen the model's accuracy somewhat. For example, using base-specific order and ship times will violate the binomial backorder allocation assumption (each base could have a different shipping lead time, therefore the depot backorder status would be dependent on when shipping must take place). This problem only affects the variance calculations and dissipates with a small number of backorders or a large number of bases (Sherbrooke, 2004: 112).

DRIVE

In 1995, Louis Miller with the RAND Corporation developed an alternative approach to METRIC–based stock leveling. His method was rooted in DRIVE (Distribution and Repair in Variable Environments), a program that was originally intended to provide prioritization for depot repair and distribution (Miller, 1995: iii).

This differs with respect to METRIC-based approaches in several ways. Most importantly, METRIC-based approaches have objective functions that minimize base EBOs, whereas DRIVE's objective is to repair and distribute parts to maximize the probability that all bases meet their availability goal at the end of a planning horizon. Also, DRIVE's logic assumes that item cannibalization is performed by assuming that broken parts are consolidated on as few aircraft as possible at each location (Miller, 1995: 29).

Miller compared DRIVE leveling against METRIC leveling and indicated that the performances were similar (Miller, 1995: 40 & 68). VARI-METRIC is known to be superior to METRIC (Sherbrooke, 2004: 102) and if either METRIC or VARI-METRIC were to also implement cannibalization, it would almost certainly gain an advantage to DRIVE.

Additionally, DRIVE is better suited for issuing stock to bases rather than determining an appropriate quantity to retain at the depots (Reynolds, 1995: 11). As a result, further study of the DRIVE-based solution was not pursued. Miller's mathematics are not shown here but can be found in the literature (Miller, 1992: 16-19).

Cannibalization

Although this particular topic was not investigated in this thesis, a brief discussion of relevant cannibalization research is included to give a further understanding of the DRIVE model, as well as to serve as a basis for continued student research emanating from this endeavor.

Cannibalization is the process of taking parts from an inoperable aircraft to install on another aircraft to increase the total number of mission capable planes. It is particularly attractive because it increases the effectiveness of the inventory policy and minimizes the monetary investment required for a given level of warfighter support (Abell, 1993: xii). However, there is a maintenance cost associated with the practice that may ultimately act as detriment to aircraft availability (Fisher, 1989: 154).

Donald P. Gaver, of the RAND Corporation, wrote a detailed cannibalization model in 1993 for LRUs. A description of his work follows.

If one assumes that there are n types of LRUs on a weapon system, that LRU_i is a particular type of LRU, and that there are q_i of LRU_i installed on each aircraft, and B_i represents a backorder for LRU_i, then no more than k aircraft can be down at any time if cannibalization is performed to its maximum extent. This is shown in the following equation (Gaver, 1993: 14):

$$B_i \leq kq_i \tag{2.27}$$

Further, letting D_c be the number of aircraft for cannibalizable parts and X_i be the number of LRU_i in the pipeline, the probability distribution of the number of aircraft non-operational because they could not be fixed through cannibalization is (Gaver, 1993: 15):

$$P\{D_c \le k\} = \prod_{i=1}^{n} P\{X_i \le kq_i + s_i\}, \quad k = 0,\ 1,\ 2,\ ...,\ a \tag{2.28}$$

Of course, planes can also be non-operational for broken parts that are non-cannibalizable; the number of planes fitting this criteria is D_{nc}. Let the total number of non-operational planes be $D^{\#}$. Because of cannibalization, $D^{\#}$ must be equal to the larger of D_{nc} and D_c. If $\underline{s}$ is defined as the vector of all stock levels for a type of plane, then through independence on the ability to cannibalize different types of parts (Gaver, 1993: 16):

$$P\{D^{\#} \le l \mid \underline{s}\} = P\{D_{nc} \le l \mid \underline{s}\} \cdot P\{D_c \le l \mid \underline{s}\} \tag{2.29}$$

If there are a planes, then the expected number down is (Gaver, 1993: 16):

$$E(D^{\#} \mid \underline{s}) = a - \sum_{l=0}^{a} P\{D_{nc} \le l \mid \underline{s}\} \cdot P\{D_c \le l \mid \underline{s}\} \tag{2.30}$$

Consequently, the expected number of planes operational, or the expected availability is (Gaver, 1993: 16):

$$E(A \mid \underline{s}) = \sum_{l=0}^{a} P\{D_{nc} \le l \mid \underline{s}\} \cdot P\{D_c \le l \mid \underline{s}\} \tag{2.31}$$

Lateral Resupply

Lateral resupply is the process of a base, centralized intermediate repair facility (CIRF) or depot procuring a part from another location of the same echelon, and could perhaps be extended to mean any order and shipment that does not move a serviceable part to a lower echelon.

Sherbrooke states that lateral resupply has resulted in backorder reduction of more than 70% for low demand items in simulations when repair is restricted to the depot (Sherbrooke, 2004: 245). His work towards developing an analytic model for base repairable items was unsuccessful and was not documented (Sherbrooke, 2004: 257).

Business rules are often employed to determine from where the base backorder is resupplied. For example, one set of rules found in the literature is (Lee, 1987: 1305):

1) *Random source rule. The source base is chosen randomly from among members with stock on hand.*
2) *Priority source rule 1. The base with the maximum stock on hand is chosen as the source base. Ties are broken randomly.*
3) *Priority source rule 2. The base with the maximum stock on hand is chosen as the source base. If there are ties, then the base with the fewest backorders is chosen. Ties are broken randomly.*

The calculations for Lee's method require that bases capable of lateral resupply have identical demand rates. This assumption is not practical for USAF logistics applications.

Another approach is to create *pooling mechanisms*. These are rules that dictate how locations share resources to simplify or attempt to eliminate the lateral resupply problem. Cohen defines pools as (Cohen, 1986: 25-28):

1) *Each group is disjoint.*
2) *The collection of groups form a covering of the set of locations at each echelon.*
3) *All members of a group have a single antecedent at the next highest echelon.*
4) *The intragroup emergency shipment cost is less than the emergency cost from a location outside of the group.*
5) *Pooling within a group is controlled by a sharing rate parameter.*

Heuristics have been developed to make an economic case for when lateral resupply is beneficial (Evers, 2001, 312). The Air Force is only secondarily concerned with the cost of transportation in the case of lateral resupply. The deeper issue is how to

construct stock levels (subject to a budget) given that some parts benefit more from lateral resupply than others.

Simulation-Based Leveling

Simulation based leveling on the industrial scale has only recently become computationally possible. There is very little published research in this area; the large majority of related articles focus on the multi-echelon aspects of traditional supply chain relationships, typically modeling resource attributes such as utilization, queue length and shipping time variances. Additionally, industry generally uses normally distributed random variables; this assumption has undesirable consequences for AF supply chain modeling (Chakravorty, 1992: 165). Due to limited applicability, these articles will not be discussed further here.

One notable body of research, by Roberto Carlos Borges de Abreu, used Arena-based simulation to calculate EBOs that were numerically similar to METRIC. His method, called P-METRIC, uses the gamma distribution to model the number of items in the reparable asset pipeline (Abreu, 2002: 46). The majority of Abreu's research focused on describing the sensitivity of his model relative to changes in input parameters.

He did not argue that his method was an improvement to the traditional methods of EBO calculation and did not describe a process to use his simulation model to allocate spares. Further, his choice of the gamma distribution was because of its ability to change shape with adjustments to its shape and scale parameters; optimizing these parameters may have distorted or obscured important output information (Abreu, 2002: 50).

Investigations of the impacts of cannibalization and lateral resupply were deliberately avoided (Abreu, 2002: 43).

Another unpublished article describes how simulation's natural variability interferes with the marginal analysis process (Slay, 2007: 1). To overcome this issue, Slay recommends a trace-driven (externally supplied discrete event list) simulation and develops a novel variance reduction technique he refers to as the Distribution Enforcement Method.

Slay acknowledges that use of common random numbers, or other variance reduction techniques have been shown to improve simulation results in some circumstances, but still do not achieve the desired effect in a reasonable amount of time. The Distribution Enforcement Method (DEM) uniformly assigns event times to all demands that occur within all replications of a simulation. It then assigns each of these events to a particular replication such that the variance of demands occurring amongst replications is consistent with the variance of the demand distribution, and the mean number of demands per replication is correct (Slay, 2007: 2-7).

He notes that this method may be used to address currently unsolved analytic problems such as describing the impacts of lateral resupply, prioritized repair, expedited resupply and optimizing spares by when sorties are flown (Slay, 2007: 9). Additionally, steady state determination and base-depot interactions are left as future research issues (Slay, 2007: 11).

Methodology

Chapter Overview

This chapter outlines the research plan and provides a description of the analysis tools and methods used for model verification. The modeling process will take place in three phases shown in Figure 3. Each phase will build upon the previous one. The result will be a simulation model capable of building optimal stock levels and estimating the associated base backorders. This model will be the basis for investigating implications of Sherbrooke's modeling assumptions, such as insignificant lateral resupply.

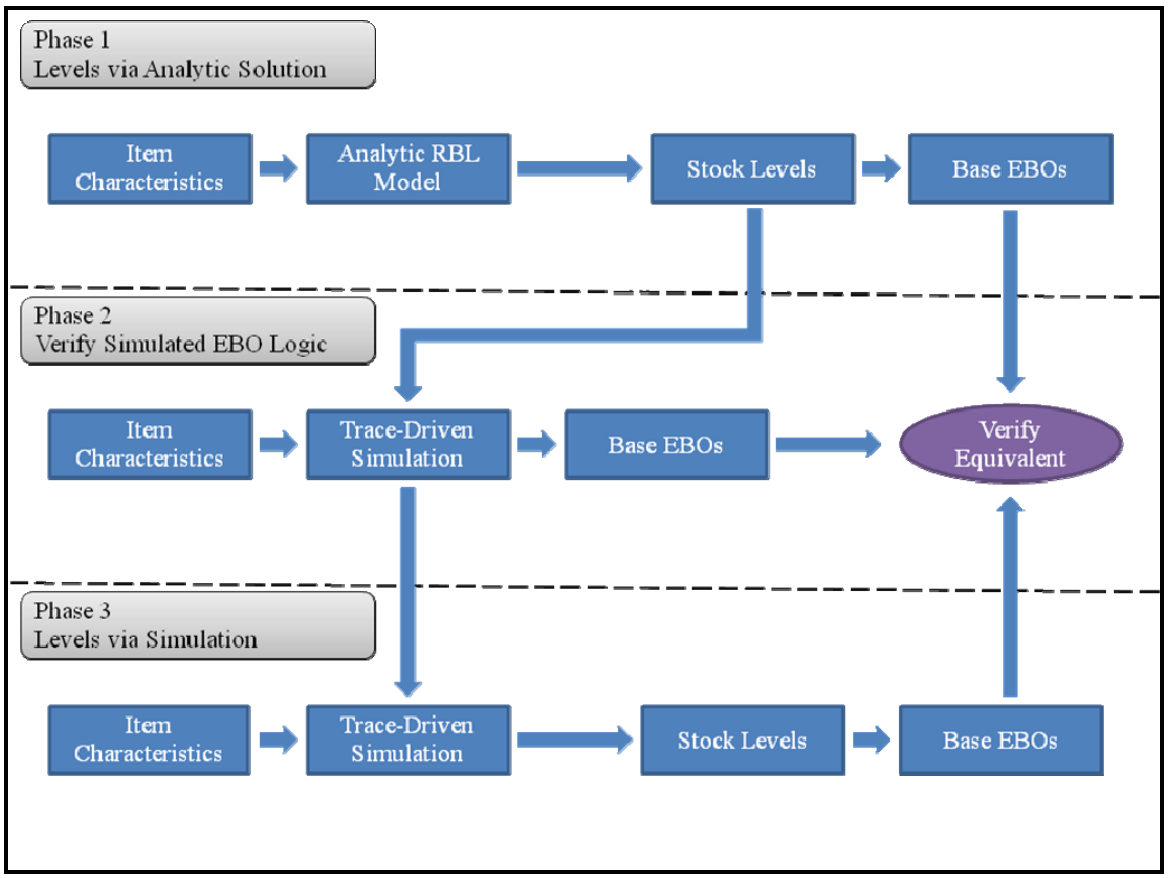

Figure 3. Model Verification Concept

The first phase will be to build a functional readiness-based sparing (RBS) model. This model will consist of a depot and five bases using METRIC logic to optimize sparing, and will result in levels with corresponding expected backorders (EBOs).

The second phase will be a trace-driven simulation to record observed backorders (an estimate of EBOs) at a single depot and five bases. The results of the trace-driven simulation and RBS model will be compared. This step will verify that simulated backorder estimation is being done correctly. Ideal simulation settings (for high accuracy and low run time) will be found and the Distribution Enforcement Method will be tested (Slay, 2007).

The third phase will be to modify the trace-driven simulation to perform readiness based sparing; the simulated levels will result in observed backorders that will be compared with the theoretical estimate from the analytic RBS solution.

All modeling, analytic and simulation-based, will be conducted in MATLAB. Although MATLAB is not typically regarded as a premier simulation package, its flexibility and debugging features make it ideal for this type of research.

Phase 1 – Analytic RBL Model

METRIC-based solutions are currently used in the majority of US Air Force models. The Aircraft Availability Model (AAM) calculates stock levels for worldwide requirements. The pipeline segments for all bases are added together for each part. The AAM uses marginal analysis to determine how many of each part is required to maximize

aircraft availability; the stopping rule is a budgetary constraint. These worldwide levels are then passed to RBL.

RBL takes the worldwide levels and divides them amongst locations with potentially unique pipelines. It accomplishes this by minimizing worldwide EBOs.

If marginal analysis is used to allocate levels instead of a total enumeration of all solutions, the resulting levels can be suboptimal; Sherbrooke mentions when identical bases exist, the marginal analysis solution may result in excessive stock being placed at the depot (Sherbrooke, 2004: 53). He recommends a *flushout* procedure to assess the benefits of removing a unit of stock from the depot and placing it at a base instead.

This project ultimately generates stock levels using a simulation technique. To verify the simulation model is correct, it must be compared against the analytic procedure.

Subsequently, a MATLAB version of the AAM and RBL was built and was augmented with a *flushout* routine. This model is able to generate appropriate levels for a system consisting of one depot and five bases with accurate EBO estimation. Although the literature's examples were constructed using a single aircraft part, the MATLAB model was built to accommodate any number of parts or bases. Further details of this model will be discussed in the Results chapter.

Phase 2 – Trace-Driven Simulated Backorders

This phase focuses on the development of logic for a rudimentary simulation model that mimics the performance of a METRIC-based (analytic) RBL solution. As

pointed out by F. Michael Slay (Slay, 2007), simulation replications for this type of application can be computationally expensive; bias and variance may need to be controlled to maintain adequate fidelity in a reasonable amount of time; he argues that distribution enforcement is superior to other variance reduction techniques.

The Distribution Enforcement Method (Slay, 2007) described in the literature review was also coded into MATLAB. This method provides an event list for all replications, where an event is defined as the demand for a specific type of part. If each replication is meant to represent a possible future time horizon, and the number of demands occurring in the horizon are assumed to be Poisson with a specified demand mean, then over a large number of replications (or horizons), the percent of replications containing *x* demands should be equal to the Poisson probability *P(x)*.

The DEM MATLAB code permits the user to select random event-replication assignment in lieu of strict Poisson enforcement; this reduces the run-time of the program dramatically, but will not represent the Poisson distribution as accurately. The event-replication assignment was verified by comparing the output's histograms with the Poisson probability distribution function as shown in Figure 4. The DEM does enforce the Poisson distribution, as expected. Random assignment also approximated the Poisson distribution reasonably well in nearly one-tenth the time; however, precision may be very important to the spares allocation process.

Because there are a predetermined number of micro-replications (number of trials), and there is a constant probability that a single demand is assigned to a replication (the reciprocal of the number of micro-replications), the demands will be distributed binomially when no distribution is enforced. With a large number of micro-replications,

the binomial distribution will approximate the Poisson distribution very closely (Slay, 2007: 4). At this point, it is uncertain how much benefit distribution enforcement can provide relative to its much longer computation time (as employed in this study).

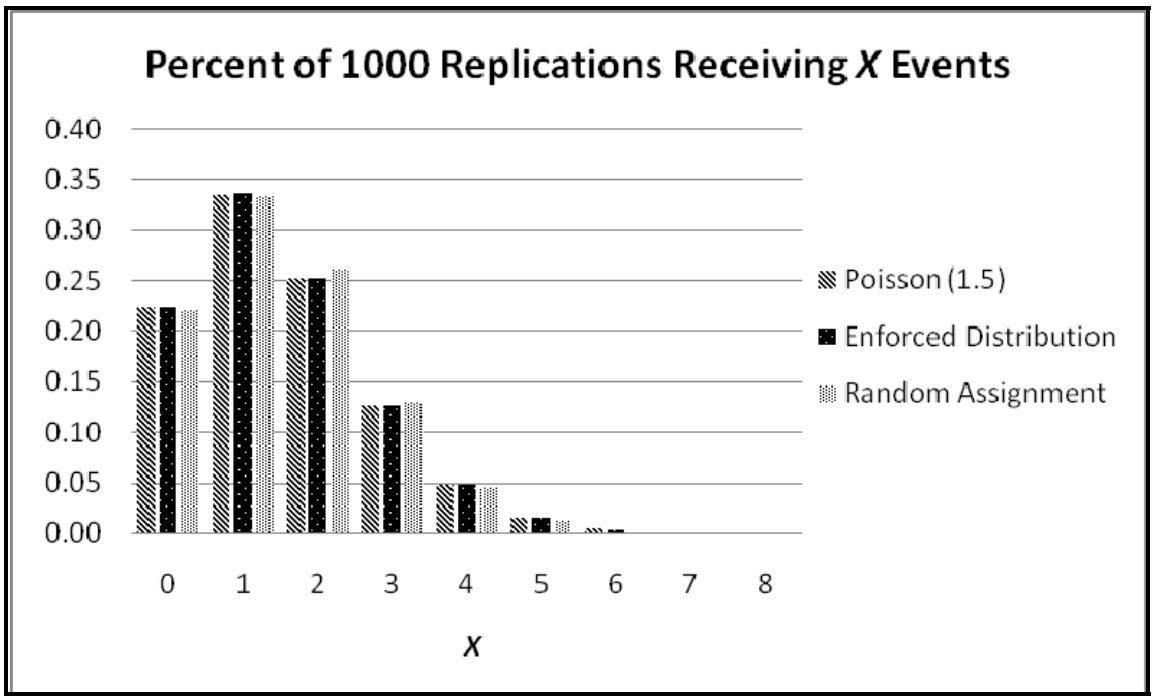

Figure 4. Percent of 1000 Replications Receiving *X* Events

Because the Distribution Enforcement Method (DEM) was originally described for a single location, the method was modified to subdivide the events by base according to one of two different methods: relative local demand rates via an inverse transform function (randomly), or alternatively, by expectation. The Bernoulli splitting principle permits a Poisson process to be subdivided into lesser Poisson processes.

The inverse transform method builds a cumulative distribution function from the proportion of each base's demand rates relative to the worldwide demand. Random numbers are used to determine which base is assigned any given demand.

The expectation method operates somewhat differently. Because each base has a demand rate and a finite number of demands will be assigned, there is an expected number of demands that will be assigned to each base. The base with the lowest percent of assigned demands already allocated to it will receive the next demand. This process continues until all demands (events) are assigned. Due to the nature of this process, events will be dispersed evenly throughout time at each location.

The two event-base assignment rules were investigated for a single part over one-thousand replications with dissimilar bases; results are shown in Figure 5.

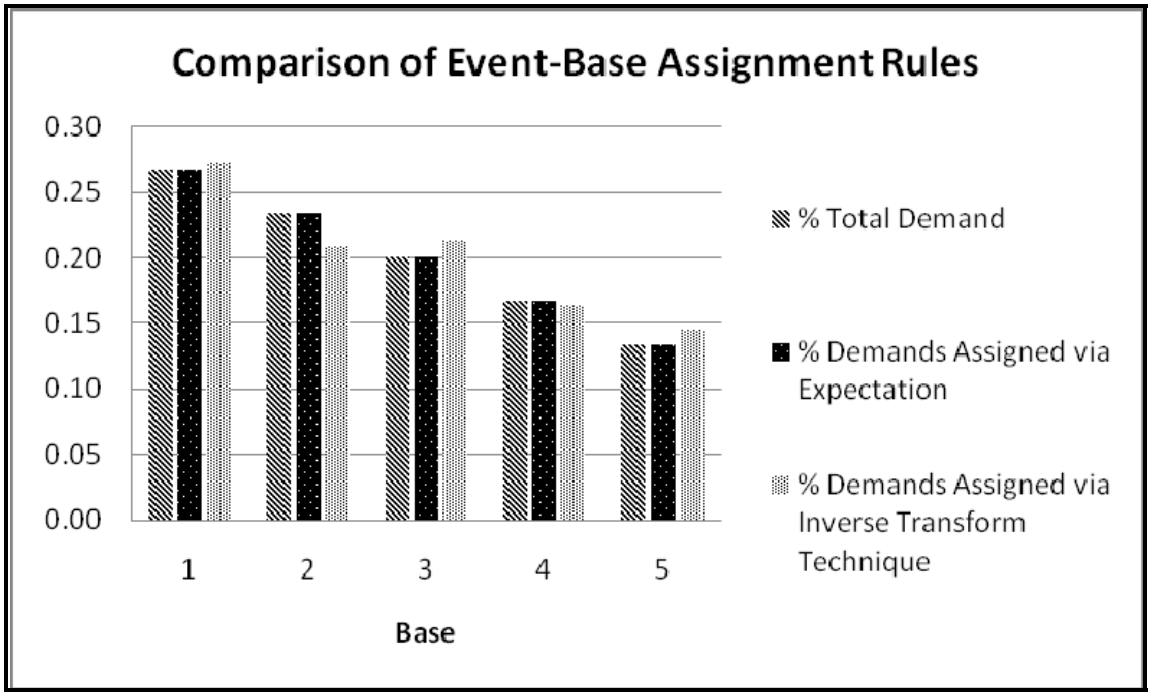

Figure 5. Comparison of Event-Base Assignment Rules

Assignment by expectation performed slightly better than the inverse transform technique. However, in the real world, demands may not always occur at a precise period as the expectation rule dictates. Some randomness may be desirable to enhance the believability of the model. This feature may prove useful in testing model robustness.

Some variation of the DEM technique will be used to generate an event list that contains at a minimum: the part type, event time, failure location, and replication number. This discrete event list provides failure data for the trace-driven simulation. The event list is the primary random feature of the simulation; the only other instances of randomness occur to break ties between minima or maxima when necessary, or to make repair source decisions with respect to the NRTS rate.

At this point, this chapter has discussed how to analytically build stock levels, and construct an event list. All of this will be instrumental in testing portions of simulation code designed to model the USAF supply system and record observed backorders.

Before the simulation begins, the event list only consists of part failure events. As parts fail, base stock is decremented; if the number of failures exceeds the number of parts the base has on hand, a backorder is generated. If a base backorder occurs, or a part is sent to the depot for repair, the base will request a part immediately from the depot. The depot must comply if it is able to do so. If a part is repaired at the base, it can be retained or sent to the depot; this decision will be based on the largest reduction in system EBOs (per METRIC logic). When a depot repairs a part, it will be used to fill an existing backorder at a location with insufficient due-in assets; in the event there are no such backorders, the depot will optimally place the asset to minimize system EBOs using METRIC logic. This process (with key simulation modeling assumptions) is outlined in Figure 6.

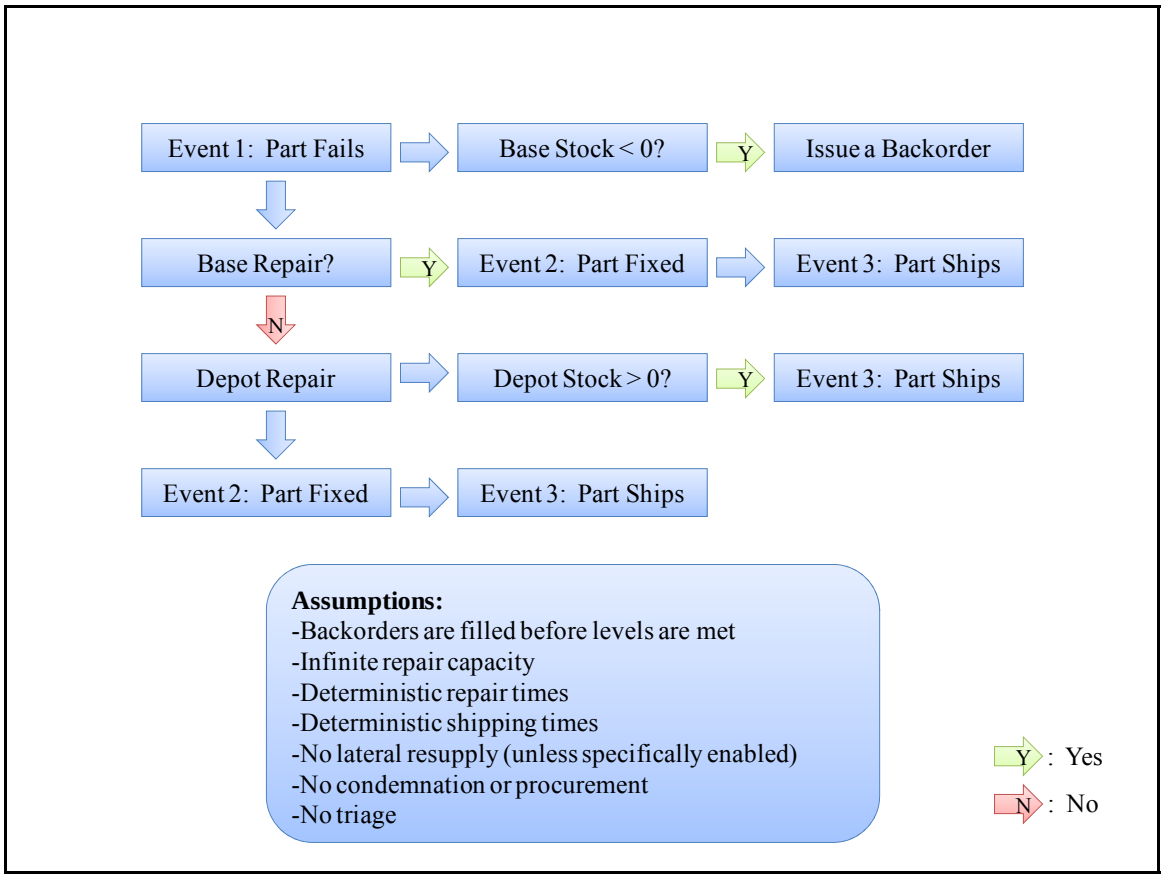

Figure 6. Trace-Driven Simulation Flow Diagram

As the simulation iterates through failures, new events are generated to represent the repair of a part or the arrival of a part at a destination. Naturally, all of these events have times associated with them, so the event list must be resorted after the completion of any event. Stock classifications such as *on hand*, *due in*, *work in process*, and *backorders* are tracked meticulously throughout the simulation. The origination and termination times for all backorders are recorded and tallied for each part type and location.

The verification of model logic for this phase will include analysis for determining steady state conditions, base repair logic, depot repair logic and depot distribution logic. It will also be important to determine appropriate run parameters for the simulation. All of this will be discussed in detail in the Results chapter.

Phase 3 – Simulated Readiness-Based Leveling

The first phase used the analytic model to build levels that minimized EBOs. The second phase took these stock levels and a list of part failures over a time horizon, and simulated the USAF supply pipeline to count the number of backorders occurring at multiple locations. These backorders were shown to correspond to the theoretical values predicted by the analytic model.

The third phase uses the simulation logic from phase 2 without the stock levels supplied in phase 1. Instead, the simulation begins with no stock being assigned to any location and a full budget. It is assumed parts are bought individually.

The simulation manages two types of stock levels: temporary and permanent. Permanent stock levels are those resulting from a *buy* decision. Temporary stock levels are those used to determine which levels should be adjusted permanently. For example, when the simulation begins, the permanent levels for all locations are set to zero. The simulation runs to establish a baseline backorder statistic. The simulation runs again for each part-location combination, temporarily incrementing each location's stock level by one unit for a specific type of part.

After a complete set of simulations (all part-location combinations have been exhausted), the reduction in backorders from the baseline statistic is divided by the appropriate part's cost. The part-location combination with the largest decrease in backorders per dollar spent is selected; one of these parts will be *bought*, and placed at that location. Thus, the permanent level and budget are updated.

Precautions are taken to ensure that a location's stock never exceeds the permanent stock level. This process continues until the budget is expended.

Simulated readiness-based leveling, in this context, cannot be totally independent from METRIC-like logic; there still must be a way for the program to determine which location a repaired asset should be sent to. However, using this approach, the rigid assumptions of METRIC can be bent, if not altogether broken. If necessary, levels can be generated with modeling considerations beyond the capability of an analytic solution such as transportation costs, lateral resupply, finite repair capacity and queuing, and maintenance or scheduling anomalies (such as triage). This capability can greatly influence the quantity of each asset that the AF elects to hold, and can provide insight into answering difficult questions.

An example is the base lateral resupply problem; it is of great interest to know when lateral resupply is beneficial, and how much it affects observed backorders. All previous attempts to model this phenomenon have failed. By adjusting the simulation logic to allow bases to fill their backorders from other bases when the depot has no stock, or by allowing a stock-depleted depot to pull assets from bases with plentiful inventory, the effects on system EBOs can be quantified. With sufficient computing power, levels can be generated with this type of logic.

An easily achievable benefit is to assess the impact of policy or procedural change on the system to enable better business decisions. An equally important benefit, although slightly more difficult to achieve, is to build analytic models assessing the impacts of these changes for use in the requirements computation; this could possibly take the form of modifying the METRIC logic when certain business rules are met, such as a part's parameters falling on a certain side of a threshold. As mentioned previously, even a

small improvement in system EBOs may have enormous financial ramifications. Building such analytic models is possible through use of this type of simulation.

Modeling Assumptions

As shown previously in Figure 6, several assumptions were necessary to manage the discrete event list and are important to consider again before moving on to the Results chapter. The simulation logic used in this research forced the depot to allocate parts to satisfy backorders before providing parts to bases to fulfill their own stock levels. All repair activities were unconstrained, meaning that repairs were independent. The time required for shipping and the repair of a part were assumed to be deterministic. Procurement was assumed to offset condemnation, and thus was not modeled. Triage was assumed to take very little time and was also not considered. Lateral resupply was not permitted in the initial model, but eventually the model was altered to permit the user to enable it. Further discussion of lateral resupply modeling will take place in the following chapter.

Results

Chapter Overview

The Methodology chapter provided descriptions of the project plan for research and verification. This chapter will build upon it by showing analysis results of the verification process and providing appropriate interpretations. Additionally, results pertaining to investigative questions will also be presented, where appropriate. The format will be similar to that of the previous chapter.

Phase 1 – Analytic RBL Model

A multi-echelon optimization example for the METRIC algorithm is provided in the literature and is shown in Figure 7 (Sherbrooke, 2004: 49-53). The corresponding MATLAB output is provided in Figure 8.

Inputs		Annual Demand Rates					
		Depot	Base 1	Base 2	Base 3	Base 4	Base 5
	Demand Rate	92.8	23.2	23.2	23.2	23.2	23.2
		Repair Times					
		Depot	Base 1	Base 2	Base 3	Base 4	Base 5
	Repair Time	0.02531	0.01	0.01	0.01	0.01	0.01
		Not Reparable This Station Rates					
		Depot	Base 1	Base 2	Base 3	Base 4	Base 5
	NRTS Rate	0	0.8	0.8	0.8	0.8	0.8
		Transportation Times					
		Depot	Base 1	Base 2	Base 3	Base 4	Base 5
	Depot	0	0.01	0.01	0.01	0.01	0.01
	Base 1	0.01	0	0.01	0.01	0.01	0.01
	Base 2	0.01	0.01	0	0.01	0.01	0.01
	Base 3	0.01	0.01	0.01	0	0.01	0.01
	Base 4	0.01	0.01	0.01	0.01	0	0.01
	Base 5	0.01	0.01	0.01	0.01	0.01	0
	Cost per Item	1					
	Budget	8					
Outputs		Stock Levels by Location					
		Depot	Base 1	Base 2	Base 3	Base 4	Base 5
	Stock	3	1	1	1	1	1
	System EBOs	0.2060					

Figure 7. Image of Analytic Model Example Data (Sherbrooke, 2004, 49-53)

```
FLUSHOUT PROCEDURE
FINAL RESULT

stocklevels =

     3     1     1     1     1     1


BEBOs =

         0    0.0412    0.0412    0.0412    0.0412    0.0412


SEBOs =

    0.2060
```

Figure 8. MATLAB Output for Analytic Model Example

According to Figure 8, the MATLAB version of the analytic model determined that the total budget was best spent purchasing eight units of a certain type of part (although there were no alternatives in this example). These eight parts were then spread across locations using the marginal analysis procedure. To guard against the *flushout* phenomena, a sub-routine verified that the RBS allocation was optimal. In this case no adjustments needed to be made, and base expected backorders (BEBOs), and total system EBOs (SEBOs) were reported. The function executed in slightly over one-tenth of a second. Results matched Sherbrooke's example exactly.

This type of verification procedure was repeated for multiple parts, differing bases, and various item characteristics. No evidence was discovered to suggest that it was dissimilar to the METRIC model proposed by Dr. Sherbrooke, being used in USAF models today.

Phase 2 – Trace-Driven Simulated Backorders

The first phase of model verification was to determine an approximate steady-state run-length. A small scenario was created assuming five identical bases, and three parts with differing demand rates: fifteen, ten, and five parts per year at each base, respectively. The NRTS rate for all parts at all locations was set to 0.5. Repair times and shipping times were all identical. The DEM function was set to use random demand-replication and demand-base assignment with one-hundred replications per simulation. The simulation was run for between one and one-hundred eighty days using intervals of one day. The average observed backorders for each trial (Figure 9) were plotted against the constant theoretical EBOs from the analytic model (Figure 10).

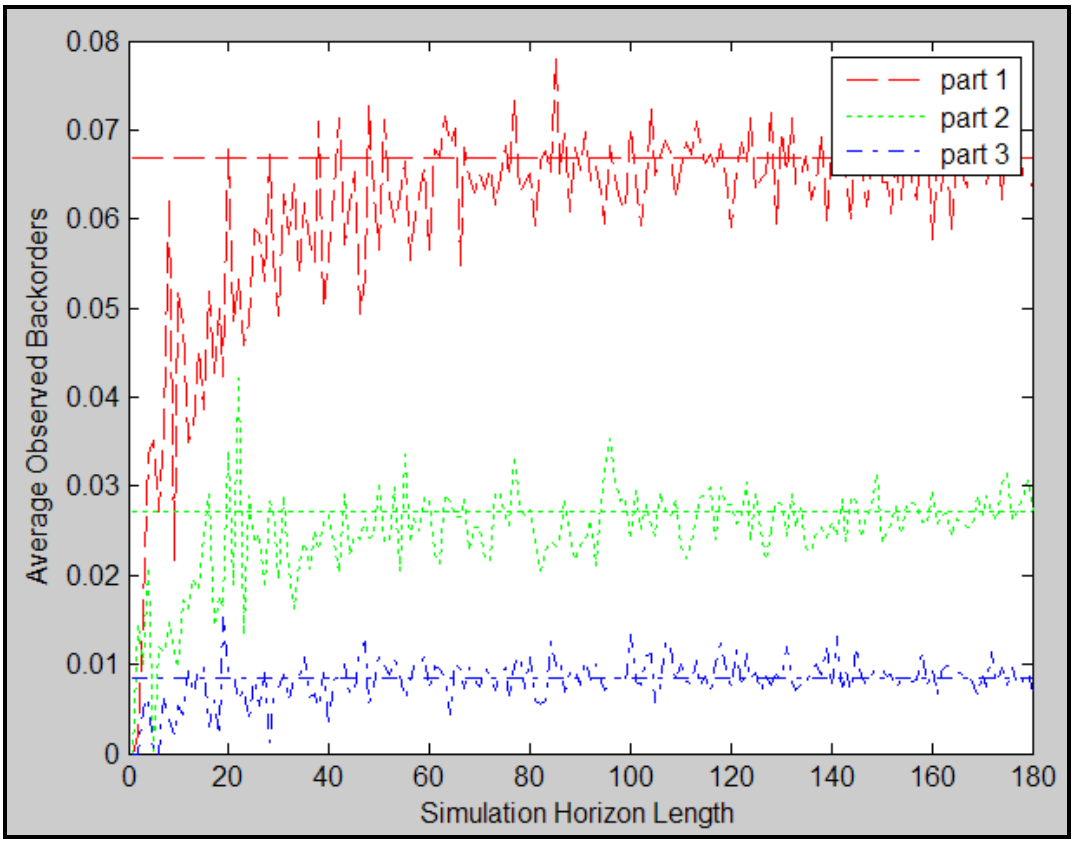

Figure 9. Observed Backorders vs. Simulated Time Horizon

```
FLUSHOUT PROCEDURE
FINAL RESULT

stocklevels =

     2     1     1     1     1     1
     2     1     1     1     1     1
     1     1     1     1     1     1


BEBOs =

         0    0.0134    0.0134    0.0134    0.0134    0.0134
         0    0.0054    0.0054    0.0054    0.0054    0.0054
         0    0.0017    0.0017    0.0017    0.0017    0.0017


SEBOs =

    0.0669
    0.0272
    0.0085
```

Figure 10. Analytic Model Output for Steady State Determination

As shown in Figure 9, the steady state conditions for this experiment seemed to occur around ninety days. The average observed backorders converged to the theoretical EBO values predicted by the analytic model, as expected. The selection of ninety days as the appropriate run-length may need to be reevaluated as parts with larger demand rates are incorporated into the model, but it serves as a valuable set point for continued verification experimentation.

In the previous experiment, the NRTS rate was set to 0.5, providing an equal balance between base and depot workload. To ensure the base logic and depot logic were working correctly (independently), the NRTS rate was set to its extreme values for runs of ninety days; this was replicated five times to form 90% confidence intervals about the means. Because each simulation consisted of one-hundred micro-replications, and the

event list was refreshed five times, there were a total of five-hundred possible futures evaluated (more discussion regarding micro-replications will take place later in this chapter).

The first trial used a NRTS rate of zero for all parts at all bases; this implies that all parts are repaired at the base (Figure 11). The second trial was for a NRTS rate of one for all parts at all bases, mandating the depot as the sole source of repair; this results in higher EBOs because inventory is unable to be used while being transported (Figure 12). This effect becomes more pronounced for parts with higher demand rates.

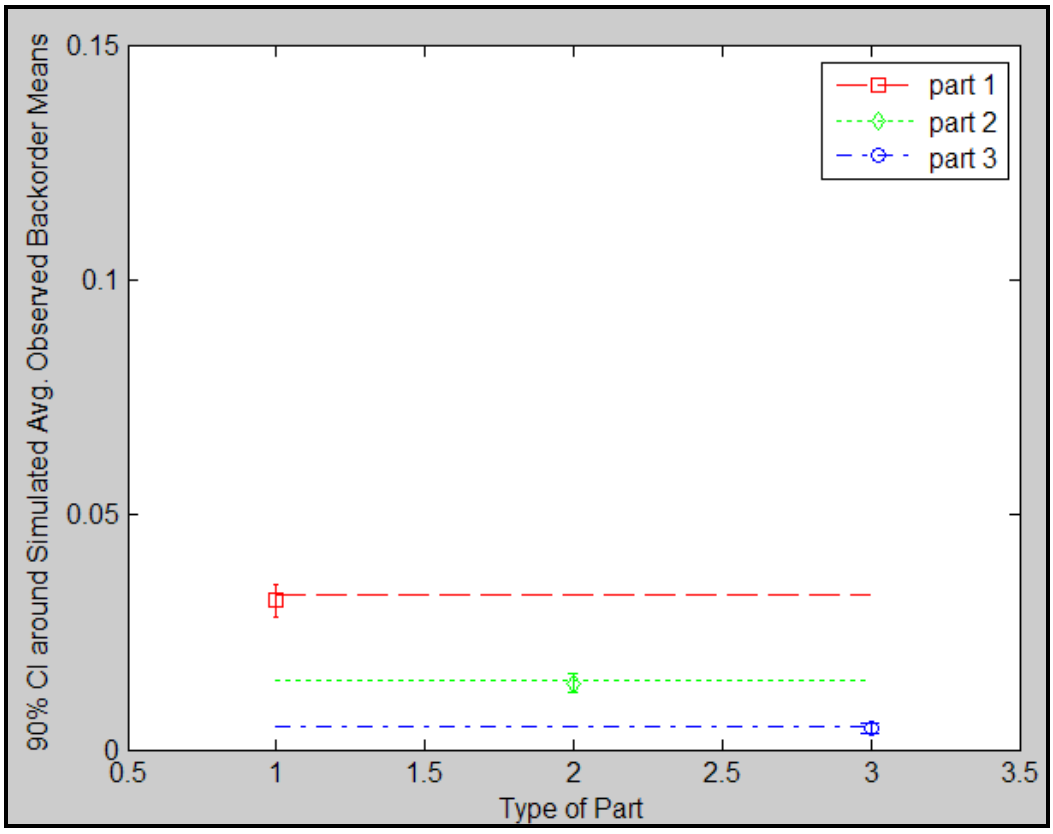

Figure 11. Average Expected Backorders for Base Repair Only

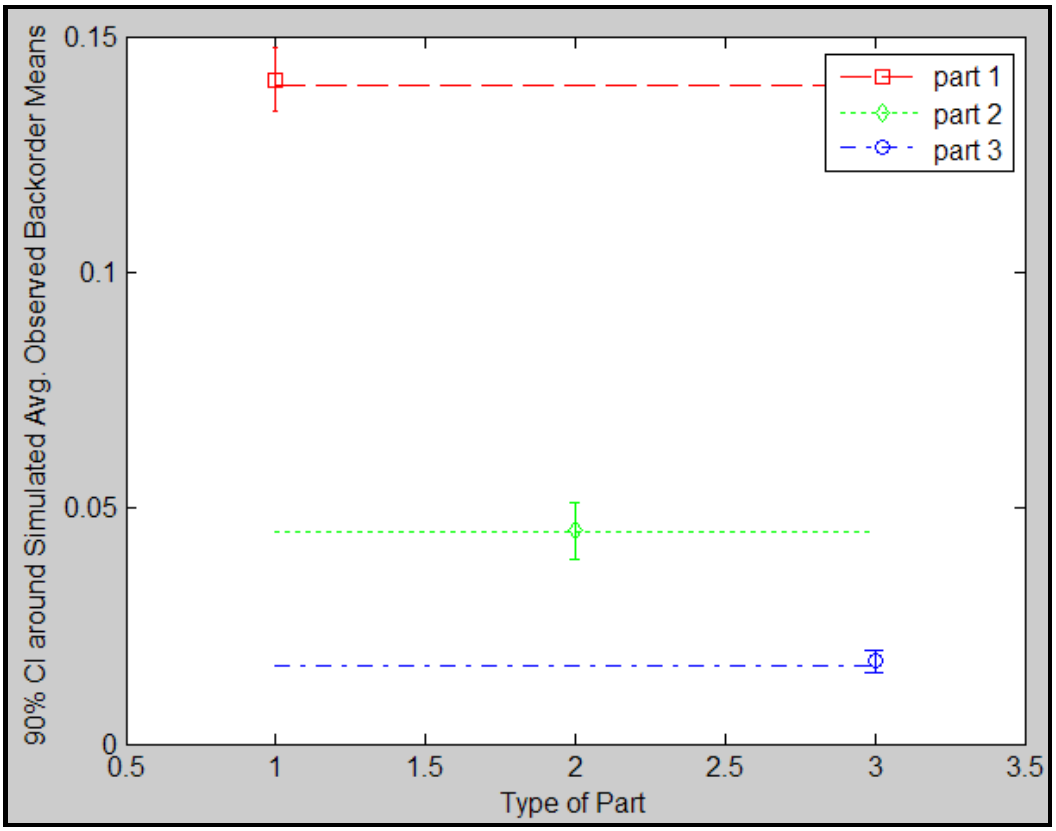

Figure 12. Average Expected Backorders for Depot Repair Only

In both figures (11 & 12), the confidence intervals for average observed backorders for all parts contained the expected backorder value from the analytic model; these experiments offer evidence that the model's base and depot repair logic is operating correctly. Furthermore, as shown in the steady state experiment (Figure 9), the interplay between the model's two repair processes seems to be correct.

So far, all verification experiments have been conducted using identical bases. Subsequently, the annual demand rates were altered to create dissimilar bases (Table 1). The model was run for five replications of ninety days with the NRTS rate set to 0.5 for all parts to generate 90% confidence intervals around the average observed backorders (Figure 13). No problems were detected.

Table 1. Annual Demand Rates for Dissimilar Bases

	Base 1	Base 2	Base 3	Base 4	Base 5
Part 1	15	16	17	18	19
Part 2	10	9	8	7	6
Part 3	7	3	7	3	5

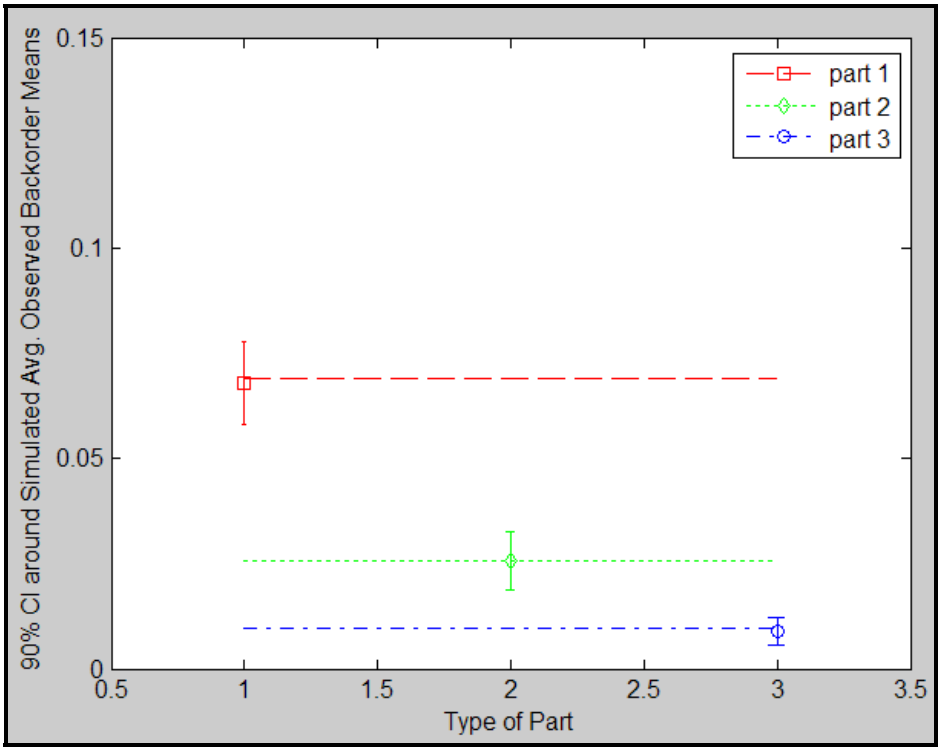

Figure 13. Average Expected Backorders for Dissimilar Bases

The simulation model was also built with several other adjustable parameters that may influence the output. These parameters are micro-replications, random *demand-base* assignment, random *demand-replication* assignment, and the option of *owner bases*.

Micro-replications refer to the number of potential futures that are tested in each run of the simulation; each future will result in a specific number of backorder-days, but over all micro-replications, an average backorder statistic can be calculated for a given run-length. Because the average backorder statistic is the number-of-interest, many

simulations (macro-replications) are necessary to construct means and confidence intervals about the statistic.

Random demand-base assignment permits the use of random numbers to be used in an inverse transform technique to allocate a demand to a base in proportion to that base's demand rate. If unselected, demands will be placed at the base with the highest expectation of receiving it. The total number of demands assigned to each base is divided by that base's expected number of demands. The base with the lowest percent of demands assigned will receive the next demand.

Random demand-replication assignment refers to the action of enforcing a demand distribution rigidly, as the DEM does (Poisson), or permitting demands to be allocated randomly amongst micro-replications (binomially).

The final parameter-of-interest is the option of using *owner-bases*. If this feature is used, a broken part that is sent to the depot for repair can only be returned to the base at which it originally failed. This concept is sometimes used for specialized engine components, but is generally not used and is unsupported by legacy systems.

To investigate the parameters' effects, a 2^4 full-factorial experimental design was used on the steady state example. Each run consisted of five macro-replications with parameter settings as shown in Table 2. The response was the sum of the absolute differences between the average observed backorders and the theoretical optima from the analytic model. The design matrix, responses and runtimes are shown in Table 3.

Table 2. Factor Labels and Experimental Levels

Parameter	Factor Label	Low	High
Micro-Replications	A	50	100
Random Demand-Base Assignment	B	on	off
Random Demand-Replication Assignment	C	on	off
Send Repairs back to Origin	D	on	off

Table 3. Factor Labels and Experimental Levels

Run	A	B	C	D	Response ($x10^2$)	Time (min)
1	1	1	1	1	1.1380	3.583
2	-1	1	1	1	0.9609	1.833
3	1	-1	1	1	0.5227	3.458
4	-1	-1	1	1	0.4249	1.725
5	1	1	-1	1	0.5212	0.583
6	-1	1	-1	1	0.9696	0.292
7	1	-1	-1	1	0.2985	0.542
8	-1	-1	-1	1	0.4195	0.250
9	1	1	1	-1	8.0290	3.333
10	-1	1	1	-1	7.8570	1.667
11	1	-1	1	-1	9.4811	3.208
12	-1	-1	1	-1	7.7469	1.625
13	1	1	-1	-1	7.8816	0.333
14	-1	1	-1	-1	8.3099	0.158
15	1	-1	-1	-1	8.3982	0.258
16	-1	-1	-1	-1	9.2381	0.117

The regression coefficient statistics for this model (Table 4) indicate that the intercept and factor D are statistically significant for an alpha of 0.05. Base ownership dramatically increases the absolute deviation from the optimal EBOs; because the objective is to minimize the response, and the coefficient for factor D is negative, it is desirable to set factor D to the high setting (or off).

It is also worth noting that the seemingly small regression coefficients are in part due to the relatively small response values.

Table 4. Regression Coefficient Statistics for the 2^4 Experimental Design

	Coef.	Standard Error	Tstat	Tcrit	p-value
Intercept	0.0451	0.0011	42.4811	2.5706	0.0000
A	0.0002	0.0011	0.2022	2.5706	0.8477
B	-0.0005	0.0011	-0.5076	2.5706	0.6333
C	0.0001	0.0011	0.0728	2.5706	0.9448
D	-0.0386	0.0011	-36.2967	2.5706	0.0000
AB	-0.0009	0.0011	-0.8231	2.5706	0.4479
AC	0.0025	0.0011	2.3647	2.5706	0.0644
AD	-0.0006	0.0011	-0.5488	2.5706	0.6067
BC	0.0003	0.0011	0.2833	2.5706	0.7883
BD	0.0029	0.0011	2.7719	2.5706	0.0393
CD	0.0010	0.0011	0.9129	2.5706	0.4032

For completeness, a regression model was fit to the statistically significant factors: the intercept, D and BD. About 99% of this data's variability was explained by this selection of terms. The regression summary statistics are provided in Table 5.

Table 5. Regression Summary Statistics for the 2^4 Experimental Design

	SS	dof	MS	Fstat	F p-value
Regression	0.0239	2	0.0120	673.5592	0.0000
Residual	0.0002	13	0.0000		
Total	0.0242	15			
R^2	0.9904				
R^2adj	0.9890				

SS: Sum of Squares
dof: Degrees of Freedom
MS: Mean Square

There are several takeaways from this exercise that will be useful in the progression of this project. For the types of experiments that have been completed so far, run-time can be reduced by using fifty micro-replications in lieu of one-hundred without sacrificing much model accuracy. Statistically, it does not matter whether demands are assigned to bases by expectation or randomly, or if demands are assigned to micro-

replications randomly or by the DEM. Because random demand-base assignment and random demand-replication assignment execute faster on a computer, these settings will be used unless otherwise specified. Finally, optimal allocation of spares shows great benefit over an *owner-base* policy as reflected by factor D.

These parameter settings will be tentatively used in Phase 3, but may need to be verified again when new part data is used or when substantial changes to the model are implemented.

Phase 3 – Simulated Readiness-Based Leveling

As mentioned previously, the objective of this phase was to replicate the output of the METRIC model using observed backorders from a simulation. By doing so, the simulation's logic can be altered in such a way as to reflect USAF operating conditions or policies that have been assumed non-existent in the METRIC model. This has the effect of eliminating bias that was previously built into the model that can improve the way the USAF allocates levels.

In phase 2, the primary thrust was equating the system observed backorders for each part type to the analytic solution's EBO estimates for model verification. Because the system backorders are the sum of the base backorders, if some base estimates were biased low, and others high, the errors would offset, and the SEBO estimate would be approximately correct. This effect aided model robustness and permitted relatively fast model run-times.

In phase 3, levels were incremented greedily one part-location combination at a time. Because decisions can never be re-evaluated, and the problem resolution has been

refined from the system level to the base level, base EBO estimation accuracy becomes more important.

Using the example resulting in the analytic model output shown in Figure 10, the run parameters found in phase 2 were used to perform simulated RBS. The results of this exercise are shown in Figure 14. As funding diminishes, the levels for each part (rows) are allocated to each location (columns), with the first column representing the depot, and each successive column representing a base. After each level is allocated, the system average observed backorders are shown.

As Figure 14 is examined, the reader note three takeaways. First, this system involves only three parts, five bases, and a single depot; nearly six minutes were required to perform each level allocation. Second, the final EBOs, shown at the bottom of the figure do not match the expected 0.1026 (from the summation of the part-specific EBOs), forecasted by the analytic model. Last, the final stock levels exactly match those shown in Figure 10.

```
Funding =

     2

Elapsed time is 328.618323 seconds.

stocklevels =

     2     1     1     1     1     1
     2     1     1     1     1     1
     1     1     0     1     1     1


ans =

    0.1899


Funding =

     1

Elapsed time is 325.125692 seconds.

stocklevels =

     2     1     1     1     1     1
     2     1     1     1     1     1
     1     1     1     1     1     1


ans =

    0.1420


Funding =

     0

Elapsed time is 326.933110 seconds.

EBOs =

    0.0879
```

Figure 14. Simulated Readiness-Based Sparing Model Output

Clearly, the time required to perform this exercise with real USAF data consisting of over 100,000 parts at hundreds of bases makes this problem infeasible on a personal computer at this time. However, this does not diminish its importance; this technique can still be used to characterize how levels would change under a hypothetical policy for different classes of parts.

Although the levels shown in Figures 10 & 14 match, coincidence is partly responsible. The accuracy of the EBO estimation was not sufficient to make this repeatable in all cases. By increasing the number of macro-replications from five to thirty, and increasing the length of the simulation from ninety to nine-hundred days, the analytic model's EBO estimate could be achieved to within 0.0003 with 95% confidence (further refinements could be possible with Slay's DEM if time were not an issue); this is a dramatic improvement over the error shown in Figure 14 that can be measured in tenths. The time to perform this technique with these new set points and current technology is prohibitive.

One could also argue that getting the right mixture of parts in the system, regardless of location is what is most important, or that differences in part-mixture in the single digits are negligible over the whole inventory system. These philosophical arguments will not be discussed further in this document.

Lateral Resupply

The simulation code used in phase 2 was modified to include lateral resupply. As the model runs, when depot stock is fully depleted and a base backorder occurs for a part

with an empty base pipeline, that base can access the on-hand stock of other bases. This process uses two steps to determine if a lateral resupply is worthwhile.

The first step is determining the base that can most afford to release the asset. Of all the bases that have at least one unit on-hand, the sum of the on-hand, due in and work in process is divided by the local demand rate. This is roughly proportional to the probability that the base will be able to fill a local demand in the near future. The base with the highest ratio will be selected as the candidate donor.

The second step is making a decision on whether it is better to wait on the depot to fill the backorder from its pipeline. The depot is checked to determine if there is a broken part currently being repaired. Because the depot has a demand rate, and the system is assumed to be in steady state, one could expect the repair rate to be equal to the demand rate. If at least one part is currently in repair, the expectation of the repair date is just half of the repair rate into the future. If half of the repair rate plus the depot-to-base shipping time is less than the base-to-base shipping time, it is not worth performing the lateral resupply.

A key reason that USAF lateral resupply research has lagged is the complex mathematics required to model this phenomena in a multi-echelon system. There are numerous inputs that may have confounding or unclear effects; it can be difficult to determine when lateral resupply is beneficial, or in some cases if it can be detrimental.

The large number of input factors and levels makes a design of experiment approach cumbersome. Instead, a two-hundred run screening experiment was performed using combinations of uniformly disturbed random settings between low and high settings. The ranges for the screening experiment are shown in Table 6.

Table 6. Lateral Resupply Factor Screening Experiment Ranges

	Low (-1)	High (1)
Annual Demand Rate	1	52
NRTS	0	1
Depot Repair Time	15 days	90 days
Base Repair Time	3 days	45 days
Order and Ship Time (base-base)	3 days	45 days
Order and Ship Time (depot-base)	3 days	45 days
Part Cost	0.5	2

It is worth noting that the verification portion of this project did not assess all values falling in these ranges, and because of certain non-linear behavior, doing so would not be an efficient use of time. Instead, it is assumed that the model may give biased EBO estimates at some values; to compensate, the response from this experiment was the percent reduction in EBOs when lateral resupply was enabled.

For each random combination of inputs, the model was run with and without lateral resupply for 730 days and twenty macro-replications subject to a budget of eight units. Other simulation set points were as described in the results at the end of phase 2. This simulation took two weeks on a recent-model laptop with a 1.8 GHz processor.

Multiple linear regression was experimented with, but no noteworthy results were found; despite numerous transformation attempts, the highest coefficient of determination found was 0.35. This is somewhat expected given the non-linear nature of the problem.

The largest observed reduction in backorders was 6.88%, and the lowest was -0.83%. Four parts had a backorder reduction greater than 2%; these parts seemed to be notably affected by lateral resupply. A two-sample t-test was performed to determine if a difference exists between the two populations; results are shown in Table 7.

Table 7. Two-Sample T-Test for Parts Affected by Lateral Resupply

	Sample Size	Mean (% EBO Reduction)	Standard Deviation	Test Data
Unaffected	196	0.06518	0.33134	
Affected	4	4.55756	1.92058	
T Statistic				-4.68
Test Statistic (α=.05)				2.35

Interesting results from the two-hundred trials are shown in Figures 15-22. The four parts with a backorder reduction exceeding 2% are labeled A-D. The horizontal axes of the graphs range from the lowest to highest observed values, and correlate to the factor settings (some factors, such as Depot Demand Rate, are functions of other facors and were not specified in terms of low and high settings).

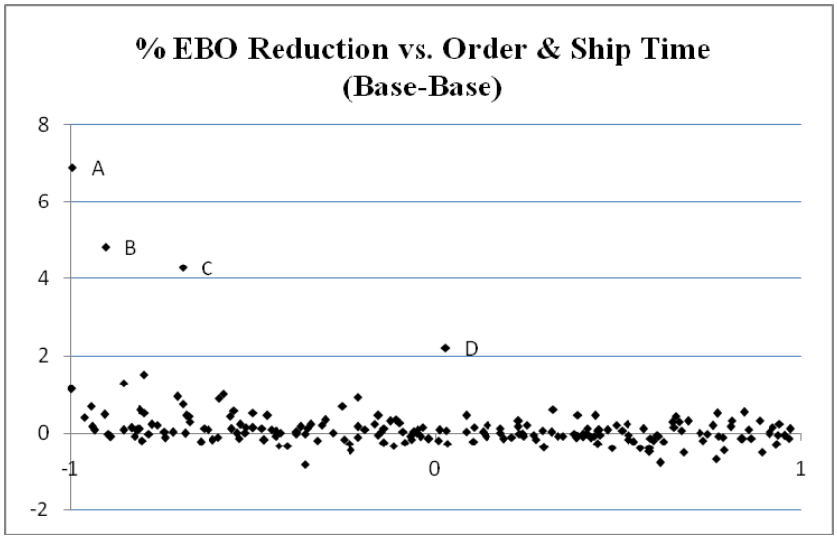

Figure 15. Percent EBO Reduction vs. Order & Ship Time (Base-Base)

The base-to-base OST seemed to have an effect. There seemed to be a reduction in backorders when bases could resupply each other quickly.

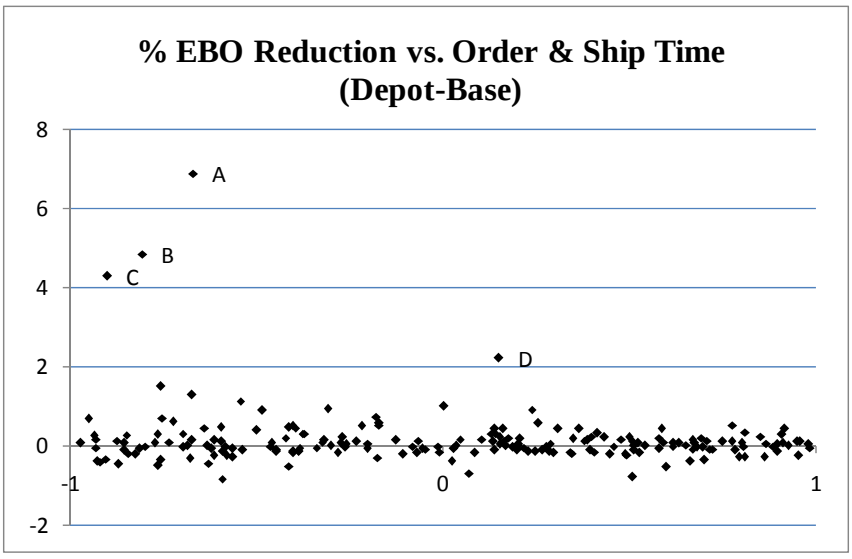

Figure 16. Percent EBO Reduction vs. Order & Ship Time (Depot-Base)

Similarly, a low depot-to-base OST may have led to a reduction of EBOs with lateral resupply enabled. Perhaps donor bases were generally replenished before backorders could occur.

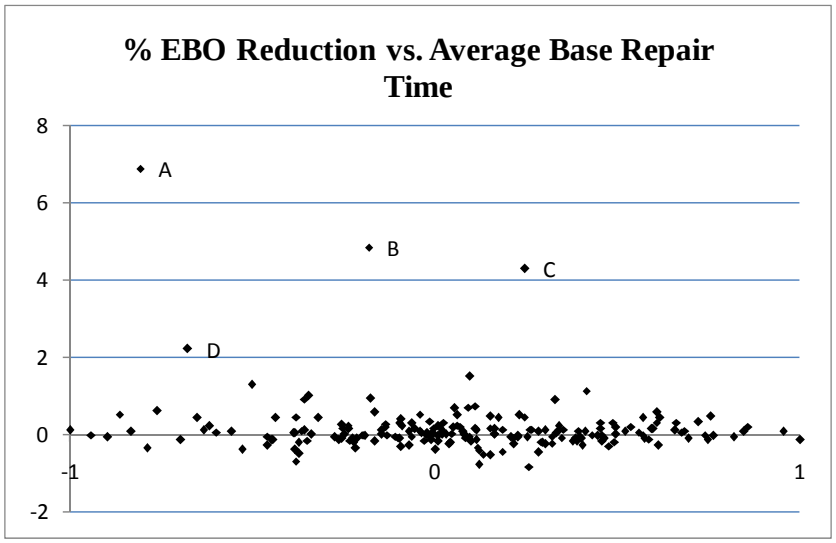

Figure 17. Percent EBO Reduction vs. Average Base Repair Time

A low average base repair time permits donor bases to recover servicible stock quickly, contributing to backorder reduction.

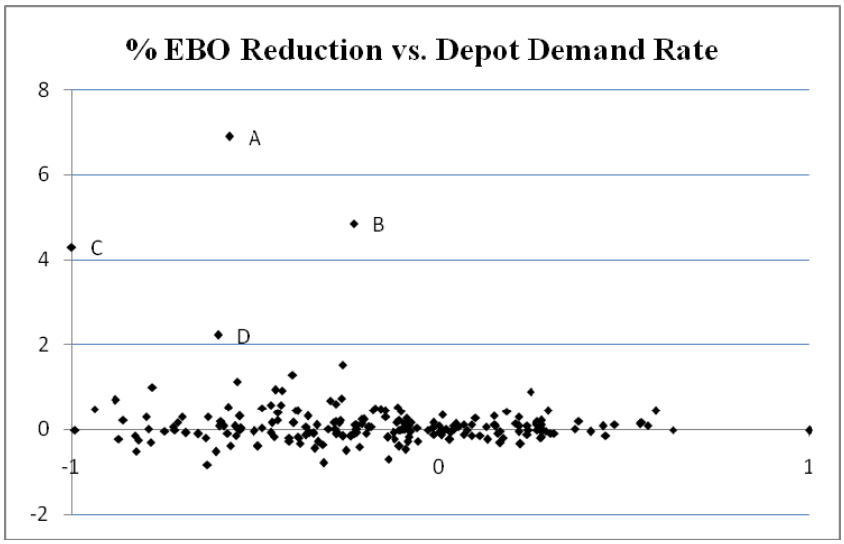

Figure 18. Percent EBO Reduction vs. Depot Demand Rate

A low depot demand rate means that more assets are housed at the bases; this may aid the donor bases in avoiding backorders. It also increases the likelihood that a base exists with on-hand assets to share with other bases.

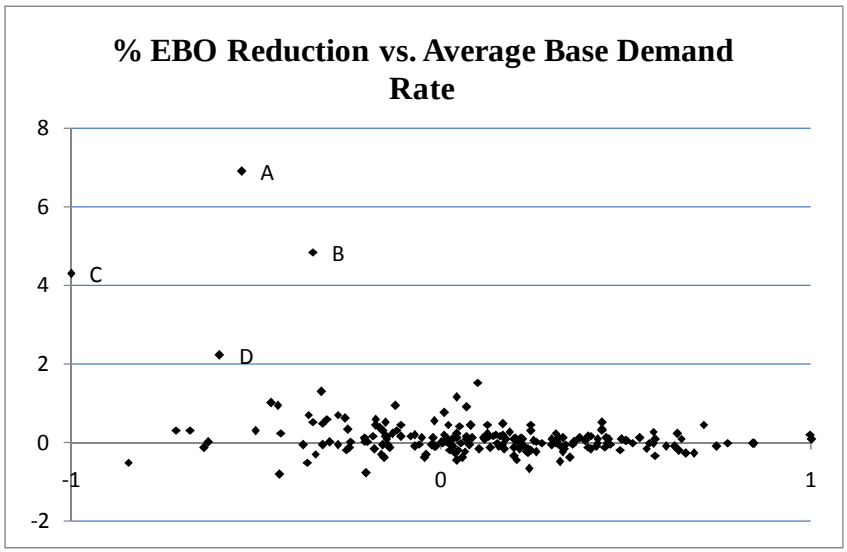

Figure 19. Percent EBO Reduction vs. Average Base Demand Rate

A low average base demand rate could impact the backorders observed at bases because demands are not able to deplete inventory.

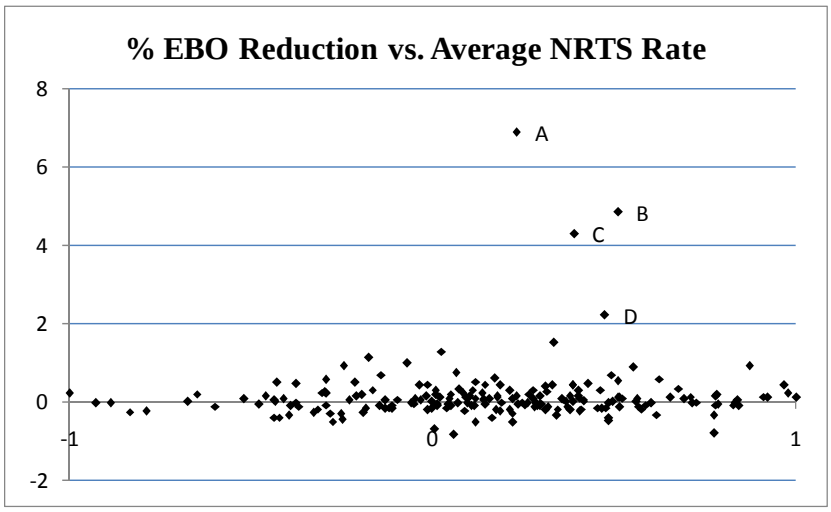

Figure 20. Percent EBO Reduction vs. Average NRTS Rate

The NRTS rate is the percent of repairs outsourced to the depot. A moderately high rate could mean that depots are filling levels at donor bases before backorders can accrue.

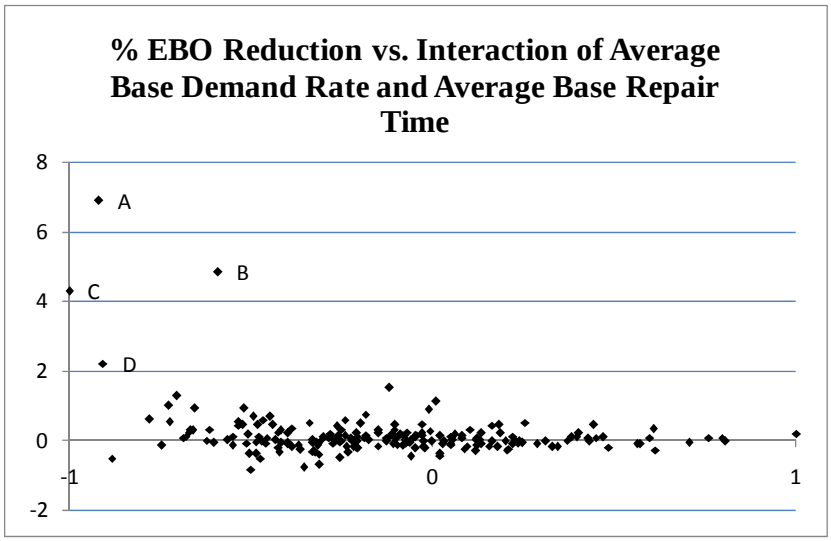

Figure 21. Percent EBO Reduction vs. Interaction of Average Base Demand Rate and Average Base Repair Time

Interaction effects were also observed. The interaction of average base demand rate and average base repair time was interesting. It appears that when parts are not

demanded frequently and can be repaired quickly, lateral resupply provides backorder reduction.

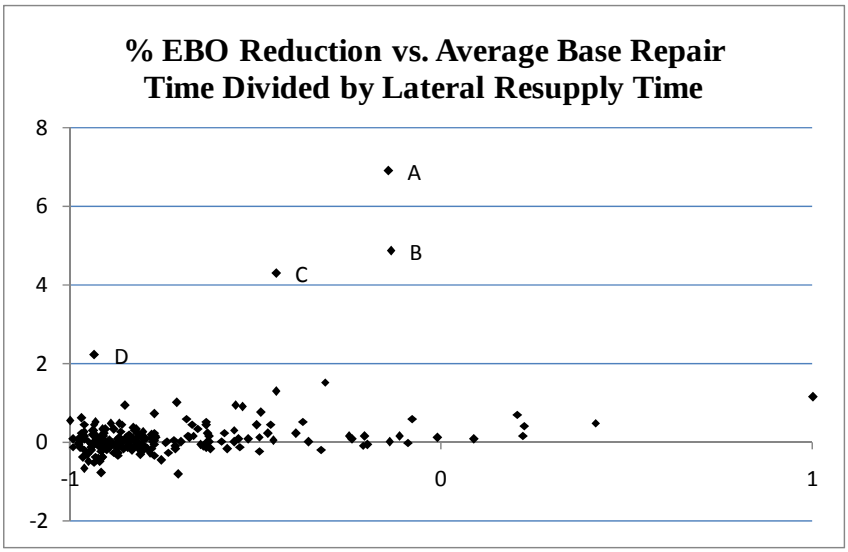

Figure 22. Percent EBO Reduction vs. Average Base Repair Time Divided by the Order & Ship Time (base-base)

Sherbrooke postulates that lateral resupply is only beneficial when the OST (base-base) is less than one-fourth of the base repair time (Sherbrooke, 2004: 258). He offers no evidence to support this claim, but may be conceptually correct; the result shown in Figure 22 seems to agree with his assertion. Lateral resupply may be beneficial if the base repair time is excessive.

Factors that failed to show a relationship in the screening experiment were the depot repair time, cost per part, and various other two-factor interactions. The data characterizing parts A-D is displayed in Table 8.

Table 8. Specific Data Characterizing Parts A-D

	A	B	C	D
Demand Rate at Base 1	17.871	9.277	8.402	26.375
Demand Rate at Base 2	11.533	40.664	3.381	24.094
Demand Rate at Base 3	19.609	28.289	3.496	1.801
Demand Rate at Base 4	3.051	3.155	5.191	4.686
Demand Rate at Base 5	19.209	9.078	5.065	8.475
Average Base Demand Rate	14.255	18.093	5.107	13.086
Standard Deviation of Base Demand	7.057	15.776	2.027	11.368
NRTS Rate at Base 1	0.766	0.275	0.267	0.460
NRTS Rate at Base 2	0.897	0.964	0.671	0.681
NRTS Rate at Base 3	0.114	0.511	0.897	0.483
NRTS Rate at Base 4	0.131	0.843	0.387	0.581
NRTS Rate at Base 5	0.762	0.526	0.705	0.856
Average NRTS Rate	0.534	0.624	0.585	0.612
Standard Deviation of NRTS Rates	0.380	0.277	0.255	0.162
Base 1 Repair Time	8.645	8.932	19.790	23.135
Base 2 Repair Time	7.873	28.808	42.052	20.386
Base 3 Repair Time	29.202	13.849	41.248	3.369
Base 4 Repair Time	3.973	13.642	21.741	6.145
Base 5 Repair Time	15.083	40.001	8.059	20.144
Average Base Repair Time	12.955	21.046	26.578	14.636
Standard Deviation of Base Repair Time	9.919	12.970	14.723	9.147
Depot Demand Rate	41.330	63.657	13.229	39.381
Depot Repair Time	42.986	34.413	88.069	75.532
Cost	0.970	1.449	0.642	1.482
Order & Ship Time (base-base)	3.095	4.991	9.424	24.565
Order & Ship Time (depot-base)	9.920	7.067	5.094	27.118
% EBO Reduction with Lateral Resupply	6.884	4.841	4.293	2.212

Using knowledge gained from the screening experiment, the factor ranges were adjusted as shown in Table 9. An additional one-hundred runs were performed to determine if a difference in response was detectable from that of the initial screening runs.

Table 9. Lateral Resupply Alternate Factor Ranges

	Low (-1)	High (1)
Annual Demand Rate	1	26 (reduced from 52)
NRTS	0.5 (increased from 0)	1
Depot Repair Time	15 days	90 days
Base Repair Time	3 days	30 days (reduced from 45)
Order and Ship Time (base-base)	3 days	24 days (reduced from 45)
Order and Ship Time (depot-base)	3 days	24 days (reduced from 45)
Part Cost	0.5	2

Upon analyzing the data, the mean benefit (reduction in EBOs) from the alternative random settings (1.33%) was found to be significantly different (α=.05) than the mean benefit from the initial random settings (0.16%). Seventeen parts (of 100) showed an EBO reduction of greater than 2%, whereas in the intial set only four parts (of 200) showed such benefit. Statistics for this experiment are shown in Table 10.

Table 10. Two-Sample T-Test for Experiments of Initial and Alt. Factor Ranges

	Sample Size	Mean (% EBO Reduction)	Standard Deviation	Test Data
Initial Ranges	200	0.15646	0.74833	
Alternate Ranges	100	1.32812	2.71995	
T Statistic				-4.23
Test Statistic (α=.05)				1.65

A more detailed examination of the contributing factors for the benefits of lateral resupply was attempted through a grant of computer time from the DoD High Performance Computing Modernization Program at Wright Patterson AFB. However, time did not permit restructuring the MATLAB code to take full advantage of parallel processing; the computing time could only be reduced by approximately 50% using serial

processing on the supercomputer. Code alteration to support the use of parallel processing could be immensely beneficial, and is left for future research.

The key takeaway from this portion of research was that lateral resupply was modeled and tested; this has never been accomplished or documented in the literature previously, despite efforts of mathematicians and logisticians over the past fifty years. Although the computer code executed slowly, future research can almost certainly increase the computational efficiency and continue the documentation of the circumstances required to observe benefits of lateral resupply. This will enable refinements of the requirements computations that will guide better allocation of Air Force sustainment funding.

Conclusion

Chapter Overview

This chapter summarizes the contributions of this research and discusses potential avenues of future research.

Summary

As stated previously, the goal of this research was to provide a flexible model of the reparable asset supply chain processes that could be modified to investigate research topics that were previously inaccessible due to mathematical complexity.

This research has produced a simulation model and documented business rules that capture the interactions resulting in the statistics predicted by Dr. Sherbrooke's METRIC model. It incorporated and tested the Distribution Enforcement Method, a novel technique for bias and variance reduction; although constrained by run time, the DEM did produce very accurate demand-replication assignments. A simulation model was used to perform the readiness-based sparing technique reserved previously only to analytic models such as METRIC. Finally, this research began to document the characterization of parts benefiting from lateral resupply.

The lateral resupply investigation was a substantial achievement. All prior attempts to model this phenomenon over the past fifty years have either failed or gone

undocumented. More research will be necessary to adequately describe the circumstances under which this policy is beneficial.

This approach to supply chain modeling does provide opportunities to improve the USAF supply chain. With appropriate time, computer resources and further research, the current requirements computation can be modified to provide better warfighter support.

Further Research

The investigative questions posed previously are only a small fraction of the breadth of questions that can be addressed with the appropriate simulation model. Simulation-based leveling will enable research of many more topics than is possible with legacy models. Future research topics are left for other students' undertaking. All MATLAB code written for this project has been included in the appendix.

Future Investigative Questions

1.) How else can lateral resupply be implemented in a simulation model? What are the benefits and detriments of doing so?

2.) Using the simulation approach provided in this document, can a detailed assessment of the conditions under which lateral resupply is beneficial be generated?

3.) How can the requirements computation be modified to make use of lateral resupply information?

4.) How can this simulation and DEM code be modified to maximize computational speed on 64-bit clusters?

5.) What are the additional effects, if any, of lateral resupply on multi-indentured parts, or those undergoing cannibalization?

6.) How can lateral resupply modeling affect post-legacy USAF supply systems?

7.) Was Slay's assertion that traditional methods of variance reduction are too slow and insufficient correct? Can other methods of variance reduction be postulated?

8.) Can a simulation model be enhanced to consider transportation options, costs and business rules for prioritization of repair and distribution?

9.) What is the impact of often-ignored supply chain characteristics such as repair capacity, cannibalization, throughput, flight line maintenance time and transportation time between bases?

10.) Can simulated Readiness-Based Sparing be performed in a more computationally efficient manner?

11.) How can a model be built to evaluate levels and effects of CIRFs?

12.) Does modeling of CIRFs and cannibalization enable more realistic projections of system availability and improved inventory solutions?

13.) Owner-bases were shown to be detrimental; how appropriate is this policy for engine components?

Appendix: MATLAB Code

BaseFlush

```
function[col]=BaseFlush(part, base, locnum, Pswitch, S0, tempddr, RTd, FT,
NRTS)
%Determine if bases are overstocked and reallocate to depots

%Find System EBOs for stock as is
basenum=locnum-1;
Pipe=pipeline(1, locnum, Pswitch, S0, tempddr(part,:), RTd(part,:), FT(1,:),
NRTS(part,:));
SEBOs=zeros(1,1);
for j=2:locnum
    SEBOs(1,1)= SEBOs(1,1)+(nbEBO(Pswitch, S0(1,j), Pipe(1,j)));
end
OldSEBOs=SEBOs;  %the base keeps the part
templevels=S0;
List=zeros(3,basenum+1); %create row vector of SEBOs
List(2,:)=1:locnum;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

templevels(1,base)=templevels(1,base)-1;  %pull one from the base
templevels(1,1)=templevels(1,1)+1;  %put it at the depot
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%find system EBOs
Pipe=pipeline(1, locnum, Pswitch, templevels, tempddr(part,:), RTd(part,:),
FT(1,:), NRTS(part,:));
SEBOs=zeros(1,1);
for j=2:locnum
    SEBOs(1,1)= SEBOs(1,1)+(nbEBO(Pswitch, templevels(1,j), Pipe(1,j)));
end
NewSEBOs=SEBOs;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if NewSEBOs(1,1)<OldSEBOs(1,1)
    col=1;
else
    col=base;
end
end
```

bosim

```
% takes in eventlist and determines base and system backorders
%requires DDR, eventlist, RT, Cost, NRTS, Pswitch, FromTo, reps, days
%specifically used in readiness based leveling via simulation

%%Event Types
% 1) Part fails at location X
% 2) Part is repaired at location X
% 3) Part arrives at base Y (Depots redistribute to bases to fill BOs)

clear Allsebos BEBOs BO BOlist DI Demandtimes E EBOsbyRep OH S SEBOs actualBO
SEBO y
clear TotalEBOs TotalError Vars WIP X Xbars basevector copytoexcel counts days
deletenum
clear enforcebases expdem findsteadystate flag interval localstock locnum nrts
obsBEBOs
clear obsSEBOs out p p1 p2 p3 randomdraw sebovect storeBO storeDDR storecol
tempEL tempDDR
clear temprate tempddr u w warning xtimes List OldSEBOs I V flush storecoldep
storecolbase
clear sebovect firstbase lastbase
close all
permlevels=stocklevels;  %this is equivalent to all zeros in the beginning
because of script in the simlevels program.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
days=730;  %should be >=90
xtimes=20;%<-----User input:  number of times program is run (macroreps)
Pswitch=1;%<-----User input:  uses Hill-Stevens Pipelines
sendback=0;%<----User input:  1 returns parts to origin,0 places them optimally
tracking=0;%<----User input:  Creates Allevents with indices; slow
%laterals1=0;%<--User input: 1 permits laterals upon base backorder,0 disables
laterals2=0; %<--User input:  1 permits laterals upon a depot backorder, a
setting of 0 disables it, if enabled, laterals1 must be enabled also
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

partnum=size(DDR,1);
locnum=size(DDR,2);
basenum=locnum-1;
DDR(:,1)=updatedepotddr(partnum, locnum, DDR, NRTS);
tempddr=DDR/365;          %<<<<<<<<<Demands per year converted into demands per
day
RTd=RT*365;               %<<<<<<<<<convert RT to days
FT=FromTo*365;            %<<<<<<<<<convert FromTo to days

if laterals2==1
    laterals1=1;
end

if xtimes==1
    neweventlist=0; %no more than one eventlist per level allocation
end

Allsebos=zeros(partnum,xtimes);
for y=1:xtimes
    if neweventlist==1
        distenf
    end

    if tracking==1
```

```
        Allevents=[];
    end

    eventlist=eventlist(:,1:4);
    eventnum=size(eventlist,1);
    days=ceil(max(eventlist(:,2)));

    %first column of tally gives number of events in the replication
    %second column of tally is the replication number
    tally=zeros(reps,2);
    tally(:,2)=1:reps;
    for i=1:size(eventlist,1)
        tally(eventlist(i,3),1)= tally(eventlist(i,3),1)+1;
    end

    eventlist=[eventlist, ones(eventnum,2),
random('unif',0,1,[size(eventlist,1),1])];

    storeBO=[];
    for r=1:reps  % r=replication number
        lastevent=0;
        bound=sum(tally((1:r),1))-tally(r,1);
        low=bound+1;
        high=bound+tally(r,1);
        tempEL=eventlist(low:high,:);

        if tracking==1
            tempEL=[tempEL,(1:size(tempEL,1))'];
        end

        OH=stocklevels; %zeros(partnum,locnum); %simultation begins 'empty and
idle'
        DI=zeros(partnum,locnum); %represents time between repair and receipt
by the base (shipping time)
        BO=zeros(partnum,locnum); %backorders by part and location
        WIP=zeros(partnum,locnum);%represents broken parts that are being
repaired
        BOlist=[];%index, part, base, startBO, stopBO

        if size(tempEL,1)>0
            while lastevent<days
                tempEL=sortrows(tempEL,2);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                if tempEL(1,6)==1  %1) Part fails at location X

                    UPDATENEEDED=0;
                    part=tempEL(1,1);
                    base=tempEL(1,4);
                    Z=size(tempEL,1);

                    %decrement base stock (OH can be negative, representing a
BO, (S=OH+DI)
                    OH(part,base)=OH(part,base)-1;

                    %if OH is neg, BO days begin accruing
                    if OH(part,base)<0
                        BO(part,base)=BO(part,base)+1;
                        newrow=[1,part,base,tempEL(1,2),days+.000001]; %sets
the stopbo out of bounds until a repair to offset it
```

```
                        BOlist=[newrow; BOlist]; %builds a BO list for each
replication
                        ind=[];
                        ind(:,1)=1:size(BOlist,1);
                        BOlist(:,1)=ind; %the first (dummy) col of BOlist is
changed into indices

                        if laterals1==1  %allows for lateral resupply when
there is a base backorder, updates the eventlist at the end of the EVENT1 code
                            if
(OH(part,1)<1)&&(OH(part,base)+DI(part,base)+WIP(part,base)<permlevels(part,bas
e)) %depot is out, and the base is allowed to get it
                                Tempbase1=LatRes(OH, DI, WIP, tempddr, FT,
basenum, part, base); %another base has it, and can ship in reasonable time
                                if Tempbase1<9999
                                    OH(part,Tempbase1)=OH(part,Tempbase1)-1;
                                    DI(part,base)=DI(part,base)+1;
                                    UPDATENEEDED=1;  %adds another row to
tempEL at the end of the section of code; can't do it here b/c it would damage
logic.
                                end
                            end
                        end
                    end

                    %determine when and where part is repaired
                    %ASSUMPTIONS: Inf rep capacity, Deterministic rep times, No
condemns, Triage time=zero
                    nr=NRTS(part,base);

                    tempEL(Z+1,:)=tempEL(1,:); %adds new event row identical to
the first
                    Z=size(tempEL,1);
                    tempEL(Z,6)=2; %the generated event will be an 'event 2'
and is the repair of the part at location X


                    if tempEL(Z,7)<nr %the part needs to be shipped to the
depot
                        tempEL(Z,2)=tempEL(Z,2)+RTd(part,1); %shipping time
FT(base,1), should be omitted (see pg 49, sherbrooke)
                        tempEL(1,5)=1;
                        tempEL(Z,5)=1; %fixer is depot
                        checkdepot=1;
                    else
                        tempEL(Z,2)=tempEL(Z,2)+RTd(part,base); %no shipping
time needed for base repair
                        tempEL(1,5)=base;
                        tempEL(Z,5)=base; %fixer is base
                        checkdepot=0;
                    end

                    WIP(part,tempEL(Z,5))=WIP(part,tempEL(Z,5))+1; %there is
now a part being fixed somewhere

                    depothadit=0;
                    if checkdepot==1  %if the base failure was NRTSd and the
depot has servicible OH, ship immediately
                        if (OH(part,1)>0)
                            OH(part,1)=OH(part,1)-1;
                            DI(part,base)=DI(part,base)+1;
```

```
                            tempEL(Z+1,:)=tempEL(1,:); %adds new event row
identical to the first
                            Z=size(tempEL,1);
                            tempEL(Z,6)=3;  %generate an 'event 3'
                            tempEL(Z,2)=tempEL(Z,2)+FT(1,base); %shipping time
                            depothadit=1;
                        end

                        %Permits base resupply of depot upon a depot backorder
(less severe than base backorder)
                        if depothadit==0  %depot did not have it for a NRTS
exchange
                            if laterals2==1

                                Tempbase2=LatRes(OH, DI, WIP, tempddr, FT,
basenum, part, 1);
                                if Tempbase2<9999
                                    OH(part,Tempbase2)=OH(part,Tempbase2)-1;
                                    DI(part,1)=DI(part,1)+1;
                                    tempEL(Z+1,:)=tempEL(1,:); %adds new event
row identical to the first
                                    Z=size(tempEL,1);
                                    tempEL(Z,6)=3;  %generate an 'event 3'
                                    tempEL(Z,2)=tempEL(Z,2)+FT(Tempbase2,1);
%shipping time
                                end
                            end
                        end
                    end

                    if UPDATENEEDED==1
                        tempEL(Z+1,:)=tempEL(1,:); %adds new event row
identical to the first
                        Z=size(tempEL,1);
                        tempEL(Z,6)=3;  %generate an 'event 3'
                        tempEL(Z,2)=tempEL(Z,2)+FT(Tempbase1,base); %shipping
time
                        UPDATENEEDED=0;
                    end

                else %its an event 2 or 3

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                    if tempEL(1,6)==2 % 2) Part was just repaired at location X
                        part=tempEL(1,1);
                        base=tempEL(1,4);  %came from
                        fixer=tempEL(1,5); %fixed at
                        Z=size(tempEL,1);

                        %generate new event
                        tempEL(Z+1,:)=tempEL(1,:); %adds new event row
identical to the first row
                        Z=size(tempEL,1);
                        tempEL(Z,6)=3; %new event is 'event 3' which is a DI

                        if tempEL(Z,5)==1  %it was repaired at the depot
                            DR=1;
                            if sendback==1  %parts are returned to where they
failed
                                tempEL(Z,2)=tempEL(Z,2)+FT(1,base);
```

```
                            else %parts are returned to fill BOs or the
"optimal" location
                                col=FillBOs(BOlist, tempEL, days, DI, WIP, Z);
                                if col>0
                                    tempEL(Z,4)=col;
                                    tempEL(Z,2)=tempEL(Z,2)+FT(1,col);
%shipping time for DI

                                else %this means there are no open backorders
for the part, still need to find base to send it to
                                    S0=OH+DI+WIP; %BOs accounted for in OH (can
be negative); Sherbrooke defines WIP as DI to an unknown location. WIP prevents
new DI.
                                    S0=S0(part,:);
                                    col=DepotFlush(part, locnum, Pswitch, S0,
tempddr, RTd, FT, NRTS, permlevels(part,:));
                                    tempEL(Z,4)=col; %ship it here
                                    tempEL(Z,2)=tempEL(Z,2)+FT(1,col);
%shipping time for DI (could be zero if it stays at depot)
                                end

                            end %end sendback or optimal

                        else %it was repaired at the base
                            %can keep the part at the base, or can ship to
depot
                            DR=0;
                            if BO(part,base)>0
                                col=base;
                            else
                                S0=OH+DI+WIP;
                                S0=S0(part,:);
                                col=BaseFlush(part, base, locnum, Pswitch, S0,
tempddr, RTd, FT, NRTS);
                            end
                            tempEL(Z,4)=col; %ship it here
                            tempEL(Z,2)=tempEL(Z,2)+FT(tempEL(Z,5),col);
%shipping time for DI (could be zero if it stays at the fixing base)

                        end %end if figuring out where it was repaired

                        WIP(part,tempEL(Z,5))=WIP(part,tempEL(Z,5))-1; %but
it's not shipped yet in the code
                        DI(part,tempEL(Z,4))=DI(part,tempEL(Z,4))+1;

                    else %it's an event 3

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                        if tempEL(1,6)==3 % 3) Part arrives at location
                            part=tempEL(1,1);
                            base=tempEL(1,4); %the base recieving the part

                            %adjust inventory
                            DI(part,base)=DI(part,base)-1;

                            %add stopbo to BOlist for oldest BO of that part
type that hasn't been filled yet
                            if OH(part,base)<0
                                BO(part,base)=BO(part,base)-1;
                                speclist=[];
```

```
                                for i=1:size(BOlist,1) %BOlist must have
elements because OH<0
                                    if (BOlist(i,2)==part) &&
(BOlist(i,5)==days+.000001) && (BOlist(i,3)==base)
                                        speclist=[speclist; BOlist(i,:)];
                                    end
                                end
                                [C, m]=min(speclist(:,4)); %C is the value, m
is the index; find earliest BO
                                rownum=speclist(m,1);
                                BOlist(rownum,5)=tempEL(1,2); %stopbo = time of
arrival

                            end
                            OH(part,base)=OH(part,base)+1;
                        end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                    end
                end

                lastevent=tempEL(1,2);
                if size(tempEL,1)==1
                    lastevent=days;
                end

                if tracking==1
                    Allevents=[Allevents;tempEL(1,:)];
                end

                tempEL(1,:)=[];
            end %while
        end %if tempEL still has rows

        storeBO=[BOlist;storeBO];  %accumulate all BOlists from all
replications
    end %for

    %compute total BOs and divide by reps and days to get avg for the
simulation
    %technically, this cuts off some BOs before they would actually be filled
    %subsequently, this gives a low biased estimate for BOs...
    %increase days or shrink ship & repair times to overcome bias
    Delta=[];
    if size(storeBO,1)>0
        Delta=storeBO(:,5)-storeBO(:,4);
    end
    storeBO=[storeBO, Delta];
    actualBO=zeros(partnum,locnum);

    for i=1:size(storeBO,1)

actualBO(storeBO(i,2),storeBO(i,3))=actualBO(storeBO(i,2),storeBO(i,3))+storeBO
(i,6); %forms a part x location matrix and adds up the deltas
    end

    BEBOs=actualBO/reps/days;  %average backorders per replication-day;
actualBO is a column vector

    SEBOs=sum(BEBOs')';
```

```
    Allsebos(:,y)=SEBOs;

end %y=1:xtimes replications

clear eventnum ans base part r e i j k RTd FT lastevent low high a ind m
eventnum tracking Delta lastevent
clear S0 S1 P0 P1 N0 N1 bound checkdepot cirfnum Z BOD to fixer lastbase newrow
rownum storeEBOspo storeEBOspz templevels tout
clear oldestbo firstbase speclist C n Pipe actualBO Row col Tstat sendback
tempbo DDRscaled
clear BEBOs BOlist Demandtimes E EBOsbyRep S storeBO SEBOs ddrtemp TotalEBOs nr
i j C
clear TotalEBOs TotalError Vars X Xbars basevector counts days deletenum
Coltemp
clear enforcebases expdem flag interval localstock nrt obsBEBOs
clear obsSEBOs out p p1 p2 p3 randomdraw sebovect storeDDR tempEL tempddr
clear temprate u w warning xtimes tempddr tally Pswitch totaldemands List
OldSEBOs I V flush SL
clear OH DI BO WIP AnalyticEBOs BEBOs Budget Pswitchc Response basenum depotnum
metric laterals permlevels

SimulatedEBOs=sum(sum(Allsebos')'/y);
clear Allsebos y
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

DepotFlush

```
function[col]=DepotFlush(part,locnum,Pswitch,S0,tempddr,RTd,FT,NRTS,permlevels)
%Determine if depots have too much stock and reallocate to bases

%Find System EBOs
basenum=locnum-1;
Pipe=pipeline(1, locnum, Pswitch, S0, tempddr(part,:), RTd(part,:), FT(1,:),
NRTS(part,:));
SEBOs=zeros(1,1);
for j=2:locnum
    SEBOs(1,1)= SEBOs(1,1)+(nbEBO(Pswitch, S0(1,j), Pipe(1,j)));
end
OldSEBOs=SEBOs;
templevels=S0;
List=zeros(3,basenum+1); %create row vector of SEBOs
List(2,:)=1:locnum;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for u=2:basenum+1  %for each of the bases, not the depot
    templevels(1,1)=templevels(1,1)-1;  %pull one from the depot
    templevels(1,u)=templevels(1,u)+1;  %put it at the base of the moment
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %find system EBOs
    Pipe=pipeline(1, locnum, Pswitch, templevels, tempddr(part,:), RTd(part,:),
FT(1,:), NRTS(part,:));
    SEBOs=zeros(1,1);
    for j=2:locnum
        SEBOs(1,1)= SEBOs(1,1)+(nbEBO(Pswitch, templevels(1,j), Pipe(1,j)));
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    List(1,u)=SEBOs;
    templevels(1,1)=templevels(1,1)+1;  %Reset
    templevels(1,u)=templevels(1,u)-1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%List(3,:
)=random('unif',0,1,[1,locnum]);
List(:,1)=[];

for i=locnum:-1:2
    if permlevels(1,i)<=S0(1,i)
        List(:,i-1)=[];
    end
end

if size(List,1)>0
    List=List';
    List=sortrows(List,3);

    [V,I]=min(List(:,1));
    coltemp=List(I,2);
    if List(I,1)<OldSEBOs(1,1)
        col=coltemp;
    else
        col=1;
    end
else
    col=1;
end

end
```

distenf

```
%Mike Slay's distribution enforcement technique
%assigns a demand to an event time, a replication and a base
%base numbers start at 2; 1 is reserved for the depot.

%takes 7 seconds for 1500 demands over 1000 replications.

%tic

%clc;
eventlist=[]; %delete former eventlist
partnum=size(DDR,1); %number of rows in DDR
basenum=size(DDR,2)-1; %number of columns in DDR, minus one (reserved for
depot)

temprate=DDR/365;   %required DDR input where DDR is in demands per year; DDR
has rows for each part, columns for each location
%code does not use FromTo or RT, no scaling necessary.

reps=50;%<-------------MODERATELY IMPORTANT, 50-150 SEEMS SUFFICIENT
enforcebases=0;%<-1 for random via inv xform, 0 for enforcement by expecation
RandomAssignment=1;%<--- 1 for random replication assignment, 0 for enforcement

for w=1:partnum
    ddr=sum(temprate(w,:))-temprate(w,1); %gives the worldwide daily demand
rate (all demands originate at bases)

    if enforcebases==1
        %Build a distribution of probability a demand occurs at a specific
base; will be used for inverse Xform technique
        ddrdist=[];
        ddrdist(1,1)=0;
        for i=2:basenum+1
            ddrdist(i,1)=ddrdist(i-1,1)+temprate(w,i)/(sum(temprate(w,:))-
temprate(w,1)); %creates column vector of cumulative probs
        end
        ddrdist=[ddrdist;1]; %adds another row that is equal to one, needed
later in a for loop
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    totaldemands=round(ddr*days*reps); %makes total demands integer
    basevector=zeros(totaldemands,2);
    basevector(:,1)=1:totaldemands;

    if totaldemands==0
        continue %go to next part without running the rest of this code:  w=w+1
    end
    interval=1/totaldemands;  %frequency of demands

    %Generate demand times and random numbers
    %1st col is for partnum eventually, 2nd is for relative time, 3rd for rep
number, 4th for rndms
    Demandtimes=ones(totaldemands,3); %generates a 'totaldemands x 4' matrix of
ones
    Demandtimes(:,3)=Demandtimes(:,3)*(reps+1); %set the third column (rep
numbers) out of bounds
    Demandtimes(1,2)=(interval/2); %initialize the first demand to half of the
interval size
    for i=2:totaldemands
        Demandtimes(i,2)=Demandtimes(i-1,2)+interval;
```

```
    end
    Demandtimes(:,2)=Demandtimes(:,2)*days; %scale the event times to the
length of the simulation
    Demandtimes(:,1)=1:totaldemands; %makes first column vector look like
[1,2,3,...,(totaldemands-1),totaldemands] to be used as indices

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    lamdat=ddr*days;
    upperbound=ceil(lamdat+4*sqrt(lamdat))+10;  %this will be used to determine
where the upper tail "stops" for the Poisson distribution.

    %format counts
    counts=zeros(upperbound+1,2); %creates a 'upperbound+1 x 2' matrix of
zeros; the +1 is to account for P(0).
    counts(:,1)=0:upperbound; %numbers the first column [0,1,2,...,(upperbound-
1), upperbound]

    %format tally
    tally=zeros(reps+1,2);
    tally(:,2)=1:(reps+1); %numbers the second column of tally
[1,2,3,...,reps,reps+1]

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Assign replications randomly pending user selection
    if RandomAssignment==1
        for i=1:totaldemands
            %generate uniform(.5, reps+.499) random number for each demand, and
round it

Demandtimes(:,3)=round(random('unif',.5,reps+.499999999,[totaldemands,1]));
        end
    else
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %-------->Assign replications w/ distribution enforcement
        demandsleft=totaldemands;
        Canassign=Demandtimes;

        while demandsleft>0

            %tallies number of demands in each replication
            tally(:,1)=zeros(reps+1,1);
            for i=1:totaldemands
                %the replication for the ith demand is found at
Demandtimes(i,3)
                %this is used as the row in tally, the demands are being added
up in the first column
                rw=Demandtimes(i,3);
                tally(rw,1)=tally(rw,1)+1;
            end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %counts the number of replications having 'counts(i,1)' demands
            % will be exported to the function: promeval
            counts(:,2)=counts(:,2)*0;
            for j=1:reps
                rw=tally(j,1)+1;
                counts(rw,2)=counts(rw,2)+1;
            end

            %promeval() returns the replicaton demand size to be promoted one
higher
            promotion=promeval(reps, counts, lamdat, upperbound);
```

```
            %isolate the replications that are promotable
            Temptally=tally(1:reps,:); %Copies both columns of tally, but not
the reps+1 elements; not promotable

            %randomize Temptally by sorting on a third column of random numbers
            Temptally(:,3)=random('unif',0,1,[size(Temptally,1),1]);
            Temptally=sortrows(Temptally,3);

            %the first time a rep is found having the right number of demands,
set Reprow equal to the rep number
            for i=1:reps
                if Temptally(i,1)==promotion

                    Reprow=Temptally(i,2);
                    break %get out of this for loop
                end
            end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %Canassign is the list of demands not assigned to a replication yet
            randomdraw=ceil(rand*size(Canassign,1));
            Drow=Canassign(randomdraw,1); %gives the index of the demand to be
assigned in Canassign and therefore Demandtimes
            Canassign(randomdraw,:)=[]; %delete it from Canassigns, so it can't
be issued again


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            Demandtimes(Drow,3)=tally(Reprow,2);  %Assign the 'Drow'th demand
the 'Reprow'th replication

            demandsleft=demandsleft-1;

        end %end while
    end %end method of assignment

    %tallies number of demands to each replication
    tally(:,1)=zeros(reps+1,1); %reinitialize the first column of tally
    for i=1:totaldemands
        rw=Demandtimes(i,3);
        tally(rw,1)= tally(rw,1)+1;
    end

    %gives number of replications having 'counts(i,1)' demands
    counts(:,2)=counts(:,2)*0; %reinitialize the second column of counts
    for j=1:reps
        R=tally(j,1)+1;
        counts(R,2)=counts(R,2)+1;
    end

    COUNTS=counts;
    clear i j k Temptally Tempdemands C m u upperbound ans Drow Reprow R;
    clear promotion demandsleft;

    %assign the partnumber to the first col
    Demandtimes(:,1)=ones(size(Demandtimes,1),1)*w; %sets the first column of
Demandtimes equal to w

    %assign a base for each demand -- the base selected should go in column 4
of Demandtimes
```

```
    if enforcebases==1
        %assign a base for each demand via inverse transform
        Demandtimes=[Demandtimes,zeros(totaldemands,1)];
        Demandtimes(:,4)=random('unif',0,1,[size(Demandtimes,1),1]);
        %if the random number falls between two probabilities, assign the base
        for i=1:totaldemands
            for j=1:size(ddrdist,1)-1
                if (Demandtimes(i,4)>ddrdist(j,1)) &&
(Demandtimes(i,4)<ddrdist(j+1,1))
                    Demandtimes(i,5)=j+1;
                    break
                end
            end
        end
        Demandtimes(:,4)=[]; %Deletes randoms, slides base numbers from the 5th
column into the 4th column
    else
        %assign a base by expectation
        X=zeros(basenum+1,5);
        X(:,1)=1:(basenum+1);
        X(:,2)=(temprate(w,:)*totaldemands/ddr)'; %gives the demands for each
location
        X(:,3)=ones(basenum+1,1); %this is necessary to assign the first demand
to the base w/ the largest DDR (vs. randomly)
        %Careful!  The resulting counts will all be off by one as a result

        %count the number of demands at each base
        for i=2:(basenum+1)
            for j=1:totaldemands
                if basevector(j,2)==i
                    X(i,3)=X(i,3)+1;
                end
            end
        end
        X(1,:)=[]; %don't need to look at depot

        for i=1:basenum
            X(i,4)=X(i,3)/X(i,2);
        end

        for k=1:totaldemands
            X(:,5)=random('unif',0,1,[basenum,1]);
            X=sortrows(X,5);

            [C, m]=min(X(:,4)); %C is the min value, m is the row index
            X(m,3)=X(m,3)+1; %update total base demands
            X(m,4)=X(m,3)/X(m,2); %update percentage
            basevector(k,2)=X(m,1);
        end
        Demandtimes=[Demandtimes,basevector(:,2)];
    end %end if

%build the event list; continuously augments eventlist with the Demandtimes
matrix for each part
eventlist=[eventlist; Demandtimes];
%w %display w
end %end for loop through parts

%sort by replication, then by time (despite the order shown in the code)
if totaldemands>0
    eventlist=sortrows(eventlist,2);
```

```
    eventlist=sortrows(eventlist,3);
end

out=eventlist;
%toc %stops the runtime clock

clear w lamdat j i firstbase depotnum ddr cirfnum ddrdist;
clear Canassign RandomAssignment ans lastbase flag d rw;
clear X Z base basevector bound checkepot col enforcebases eventnum
clear fixer high ind interval k lastevent low m newevent newrow nr out part r
clear rownum speclist storecol temprate COUNTS counts
return
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

ebosim

```
%purpose: takes in eventlist and determines base and system backorders
%requires DDR, eventlist, RT, Cost, NRTS, Pswitch, FromTo, reps, days
%precursor to bosim (which is used in readiness based leveling)

%%Event Types
% 1) Part fails at location X
% 2) Part is repaired at location X
% 3) Part arrives at base Y (Depots redistribute to bases to fill BOs)

clear Allsebos BEBOs BO BOlist DI Demandtimes E EBOsbyRep OH S SEBOs actualBO
SEBO Xbars E
clear TotalEBOs TotalError Vars WIP X Xbars basevector copytoexcel counts
deletenum
clear enforcebases eventlist expdem findsteadystate flag interval localstock
locnum nrts obsBEBOs
clear obsSEBOs out p p1 p2 p3 randomdraw reps sebovect storeBO storeDDR
storecol tempEL tempDDR
clear temprate tempddr u w warning xtimes List OldSEBOs I V flush storecoldep
storecolbase
close all
clc

storecoldep=[];
storecolbase=[];
Levelsa
permlevels=stocklevels;

partnum=size(DDR,1);
locnum=size(DDR,2);
basenum=locnum-1;
DDR(:,1)=updatedepotddr(partnum, locnum, DDR, NRTS);
tempddr=DDR/365;          %<<<<<<<<<Demands per year converted into demands per
day
RTd=RT*365;               %<<<<<<<<<convert RT to days
FT=FromTo*365;            %<<<<<<<<<convert FromTo to days


days=90;  %120 seems to work well for test cases
findsteadystate=0;    %<-----------User input, 1 creates EBO vs. replication
chart(increase xtimes), 0 does xtimes replications, each of size days
Pswitch=1;            %<-----------User input, uses Hill-Stevens Pipelines
sendback=0;           %<--------------User input:  1 returns parts to where
they broke, 0 places them optimally
tracking=0;           %<--------------User input:  Creates Allevents with
indices; slow
xtimes=5;             %<--------------User input, number of times program is
run (macroreps, or max days if findsteadystate=1)
laterals1=0;          %<--------------User input:  1 permits laterals upon a
base backorder, a setting of 0 disables it
laterals2=0;          %<--------------User input:  1 permits laterals upon a
depot backorder, a setting of 0 disables it, if enabled, laterals1 must be
enabled also

Allsebos=zeros(partnum,xtimes);
for y=1:xtimes
    tic
    if laterals2==1
        laterals1=1;
    end
    if findsteadystate==1
```

```
        days=y;
    end
    distenf


    if tracking==1
        Allevents=[];
    end

    eventlist=eventlist(:,1:4);
    eventnum=size(eventlist,1);

    days=ceil(max(eventlist(:,2)));

    %first column of tally gives number of events in the replication
    %second column of tally is the replication number
    tally=zeros(reps,2);
    tally(:,2)=1:reps;
    for i=1:size(eventlist,1)
        tally(eventlist(i,3),1)= tally(eventlist(i,3),1)+1;
    end

    eventlist=[eventlist, ones(eventnum,2),
random('unif',0,1,[size(eventlist,1),1])]; %random is only used for rts, nrts
decision

    storeBO=[];
    for r=1:reps  % r=replication number
        lastevent=0;
        bound=sum(tally((1:r),1))-tally(r,1);
        low=bound+1;
        high=bound+tally(r,1);
        tempEL=eventlist(low:high,:);

        if tracking==1
            tempEL=[tempEL,(1:size(tempEL,1))'];
        end

        S=zeros(partnum,locnum);
        OH=stocklevels; %zeros(partnum,locnum); %simultation begins 'empty and
idle'
        DI=zeros(partnum,locnum); %represents time between repair and receipt
by the base (shipping time)
        BO=zeros(partnum,locnum); %backorders by part and location
        WIP=zeros(partnum,locnum);%represents broken parts that are being
repaired
        BOlist=[];%index, part, base, startBO, stopBO

        if size(tempEL,1)>0
            while lastevent<days
                tempEL=sortrows(tempEL,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                if tempEL(1,6)==1  %1) Part fails at location X

                    UPDATENEEDED=0;
                    part=tempEL(1,1);
                    base=tempEL(1,4);
                    Z=size(tempEL,1);
```

```
                    %decrement base stock (OH can be negative, representing a
BO, (S=OH+DI)
                    OH(part,base)=OH(part,base)-1;

                    %if OH is neg, BO days begin accruing
                    if OH(part,base)<0
                        BO(part,base)=BO(part,base)+1;
                        newrow=[1,part,base,tempEL(1,2),days+.000001]; %sets
the stopbo out of bounds until a repair to offset it
                        BOlist=[newrow; BOlist]; %builds a BO list for each
replication
                        ind=[];
                        ind(:,1)=1:size(BOlist,1);
                        BOlist(:,1)=ind; %the first (dummy) col of BOlist is
changed into indices


                        if laterals1==1  %allows for lateral resupply when
there is a base backorder, updates the eventlist at the end of the EVENT1 code
                            if
(OH(part,1)<1)&&(OH(part,base)+DI(part,base)+WIP(part,base)<permlevels(part,bas
e)) %depot is out, and the base is allowed to get it
                                Tempbase1=LatRes(OH, DI, WIP, tempddr, FT,
basenum, part, base); %another base has it, and can ship in reasonable time
                                if Tempbase1<9999
                                    OH(part,Tempbase1)=OH(part,Tempbase1)-1;
                                    DI(part,base)=DI(part,base)+1;
                                    UPDATENEEDED=1;  %adds another row to
tempEL at the end of the section of code; can't do it here b/c it would damage
logic.
                                end
                            end
                        end
                    end

                    %determine when and where part is repaired
                    %ASSUMPTIONS: Inf rep capacity, Deterministic rep times, No
condemns, Triage time=zero
                    nr=NRTS(part,base);

                    tempEL(Z+1,:)=tempEL(1,:); %adds new event row identical to
the first
                    Z=size(tempEL,1);
                    tempEL(Z,6)=2; %the generated event will be an 'event 2'
and is the repair of the part at location X


                    if tempEL(Z,7)<nr %the part needs to be shipped to the
depot
                        tempEL(Z,2)=tempEL(Z,2)+RTd(part,1); %shipping time
FT(base,1), should be omitted (see pg 49, sherbrooke)
                        tempEL(1,5)=1;
                        tempEL(Z,5)=1; %fixer is depot
                        checkdepot=1;
                    else
                        tempEL(Z,2)=tempEL(Z,2)+RTd(part,base); %no shipping
time needed for base repair
                        tempEL(1,5)=base;
                        tempEL(Z,5)=base; %fixer is base
                        checkdepot=0;
                    end
```

```
                    WIP(part,tempEL(Z,5))=WIP(part,tempEL(Z,5))+1; %there is
now a part being fixed somewhere

                    depothadit=0;
                    if checkdepot==1  %if the base failure was NRTSd and the
depot has servicible OH, ship immediately
                        if (OH(part,1)>0)
                            OH(part,1)=OH(part,1)-1;
                            DI(part,base)=DI(part,base)+1;
                            tempEL(Z+1,:)=tempEL(1,:); %adds new event row
identical to the first
                            Z=size(tempEL,1);
                            tempEL(Z,6)=3;  %generate an 'event 3'
                            tempEL(Z,2)=tempEL(Z,2)+FT(1,base); %shipping time
                            depothadit=1;
                        end

                        %Permits base resupply of depot upon a depot backorder
(less severe than base backorder)
                        if depothadit==0  %depot did not have it for a NRTS
exchange
                            if laterals2==1


                                Tempbase2=LatRes(OH, DI, WIP, tempddr, FT,
basenum, part, 1);
                                if Tempbase2<9999
                                    OH(part,Tempbase2)=OH(part,Tempbase2)-1;
                                    DI(part,1)=DI(part,1)+1;
                                    tempEL(Z+1,:)=tempEL(1,:); %adds new event
row identical to the first
                                    Z=size(tempEL,1);
                                    tempEL(Z,6)=3;  %generate an 'event 3'
                                    tempEL(Z,2)=tempEL(Z,2)+FT(Tempbase2,1);
%shipping time
                                end
                            end
                        end
                    end

                    if UPDATENEEDED==1
                        tempEL(Z+1,:)=tempEL(1,:); %adds new event row
identical to the first
                        Z=size(tempEL,1);
                        tempEL(Z,6)=3;  %generate an 'event 3'
                        tempEL(Z,2)=tempEL(Z,2)+FT(Tempbase1,base); %shipping
time
                        UPDATENEEDED=0;
                    end

                else %its an event 2 or 3

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                    if tempEL(1,6)==2 % 2) Part was just repaired at location X
                        part=tempEL(1,1);
                        base=tempEL(1,4);  %came from
                        fixer=tempEL(1,5); %fixed at
                        Z=size(tempEL,1);

                        %generate new event
```

```
                        tempEL(Z+1,:)=tempEL(1,:); %adds new event row
identical to the first row
                        Z=size(tempEL,1);
                        tempEL(Z,6)=3; %new event is 'event 3' which is a DI

                        if tempEL(Z,5)==1  %it was repaired at the depot

                            if sendback==1  %parts are returned to where they
failed
                                tempEL(Z,2)=tempEL(Z,2)+FT(1,base);

                            else %parts are returned to fill BOs or the
"optimal" location
                                col=FillBOs(BOlist, tempEL, days, DI, WIP, Z);
                                if col>0
                                    tempEL(Z,4)=col;
                                    tempEL(Z,2)=tempEL(Z,2)+FT(1,col);
%shipping time for DI

                                else %this means there are no open backorders
for the part, still need to find base to send it to
                                    S0=OH+DI+WIP; %BOs accounted for in OH (can
be negative); Sherbrooke defines WIP as DI to an unknown location. WIP prevents
new DI.
                                    S0=S0(part,:);
                                    col=DepotFlush(part,locnum, Pswitch, S0,
tempddr, RTd, FT, NRTS, permlevels(part,:));
                                    tempEL(Z,4)=col; %ship it here
                                    tempEL(Z,2)=tempEL(Z,2)+FT(1,col);
%shipping time for DI (could be zero if it stays at depot)
                                end

                            end %end sendback or optimal

                        else %it was repaired at the base
                            %can keep the part at the base, or can ship to
depot

                            if BO(part,base)>0
                                col=base;
                            else
                                S0=OH+DI+WIP;
                                S0=S0(part,:);
                                col=BaseFlush(part, base, locnum, Pswitch, S0,
tempddr, RTd, FT, NRTS);
                            end
                            tempEL(Z,4)=col; %ship it here
                            tempEL(Z,2)=tempEL(Z,2)+FT(tempEL(Z,5),col);
%shipping time for DI (could be zero if it stays at the fixing base)

                        end %end if figuring out where it was repaired

                        WIP(part,tempEL(Z,5))=WIP(part,tempEL(Z,5))-1; %but
it's not shipped yet in the code
                        DI(part,tempEL(Z,4))=DI(part,tempEL(Z,4))+1;

                    else %it's an event 3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                        if tempEL(1,6)==3 % 3) Part arrives at location
                            part=tempEL(1,1);
                            base=tempEL(1,4); %the base recieving the part
```

```
                            %adjust inventory
                            DI(part,base)=DI(part,base)-1;

                            %add stopbo to BOlist for oldest BO of that part
type that hasn't been filled yet
                            if OH(part,base)<0
                                BO(part,base)=BO(part,base)-1;
                                speclist=[];
                                for i=1:size(BOlist,1) %BOlist must have
elements because OH<0
                                    if (BOlist(i,2)==part) &&
(BOlist(i,5)==days+.000001) && (BOlist(i,3)==base)
                                        speclist=[speclist; BOlist(i,:)];
                                    end
                                end
                                [C, m]=min(speclist(:,4)); %C is the value, m
is the index; find earliest BO
                                rownum=speclist(m,1);
                                BOlist(rownum,5)=tempEL(1,2); %stopbo = time of
arrival

                            end
                            OH(part,base)=OH(part,base)+1;
                        end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                    end
                end

                lastevent=tempEL(1,2);
                if size(tempEL,1)==1
                    lastevent=days;
                end

                if tracking==1
                    Allevents=[Allevents;tempEL(1,:)];
                end

                tempEL(1,:)=[];
            end %while
        end %if tempEL still has rows

        storeBO=[BOlist;storeBO];  %accumulate all BOlists from all
replications
    end %for

    %compute total BOs and divide by reps and days to get avg for the
simulation
    %technically, this cuts off some BOs before they would actually be filled
    %subsequently, this gives a low biased estimate for BOs...
    %increase days or shrink ship & repair times to overcome bias
    Delta=[];
    if size(storeBO,1)>0
        Delta=storeBO(:,5)-storeBO(:,4);
    end
    storeBO=[storeBO, Delta];
    actualBO=zeros(partnum,locnum);

    for i=1:size(storeBO,1)
```

```
actualBO(storeBO(i,2),storeBO(i,3))=actualBO(storeBO(i,2),storeBO(i,3))+storeBO
(i,6); %forms a part x location matrix and adds up the deltas
    end

    BEBOs=actualBO/reps/days;  %average backorders per replication-day

    SEBOs=sum(BEBOs')';

    Allsebos(:,y)=SEBOs;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    y
    toc
end %replications

AnalyticEBOs=TotalEBOs
SimulatedEBOs=sum(sum(Allsebos')'/y)

if findsteadystate==1
    EBOsbyRep=Allsebos';

    figure(1);
    plot(EBOsbyRep(:,1), '--r');
    ylabel('Average Observed Backorders');
    xlabel('Simulation Horizon Length');

    hold on
        plot(EBOsbyRep(:,2),':g');
    hold on
        plot(EBOsbyRep(:,3),'-.b');
    legend('part 1', 'part 2', 'part 3');
    a=xtimes;
    p1=ones(1,a)*sebovect(1,1);
    p2=ones(1,a)*sebovect(2,1);
    p3=ones(1,a)*sebovect(3,1);

    hold on
    plot(p1,'--','color','red');
    hold on
    plot(p2,':','color','green');
    hold on
    plot(p3,'-.','color','blue');
end

Tstat=tinv(.95,xtimes-1);  %using half-widths alpha=.1
Xbars=mean(Allsebos')';
Vars=var(Allsebos')';


for p=1:partnum
    E(p,1)=Tstat*sqrt(Vars(p,1))/sqrt(xtimes);
end

Response=abs(Xbars(1,1)-sebovect(1,1))+abs(Xbars(2,1)-
sebovect(2,1))+abs(Xbars(3,1)-sebovect(3,1));
copytoexcel=[Xbars,E];

figure(2);
errorbar(1,Xbars(1,1),E(1,1),'--s','color','red')
ylabel('90% CI around Simulated Avg. Observed Backorder Means');
xlabel('Type of Part');
axis([.5, 3.5, 0, .15]);
```

```
hold on
if size(Xbars,1)==3
    errorbar(2,Xbars(2,1),E(2,1),':d','color','green')
    hold on
    errorbar(3,Xbars(3,1),E(3,1),'-.o','color','blue')
    legend('part 1', 'part 2', 'part 3');
end

p1=ones(1,3)*sebovect(1,1);
if size(Xbars,1)==3
p2=ones(1,3)*sebovect(2,1);
p3=ones(1,3)*sebovect(3,1);
end

hold on
plot(p1,'--','color','red');
if size(Xbars,1)==3
hold on
plot(p2,':','color','green');
hold on
plot(p3,'-.','color','blue');
end
toc

clear eventnum ans base part r e i j k y RTd days FT lastevent low high a ind m
eventnum tracking Delta lastevent
clear S0 S1 P0 P1 N0 N1 bound checkdepot cirfnum Z BOD to fixer lastbase newrow
rownum storeEBOspo storeEBOspz templevels tout
clear oldestbo firstbase speclist C n Pipe actualBO Row col partnum basenum
Tstat sendback tempbo DDRscaled
clear Allsebos BEBOs BOlist Demandtimes  EBOsbyRep S SEBOs ddrtemp TotalEBOs nr
clear TotalEBOs TotalError Vars X  basevector counts days deletenum days Xbars
E
clear enforcebases expdem flag interval localstock locnum nrt obsBEBOs
clear obsSEBOs out p p2 p3 randomdraw reps sebovect storeDDR tempEL tempddr
clear temprate u w warning xtimes tempddr tally Pswitch Budget totaldemands
List OldSEBOs I V flush SL
```

FillBOs

```
function[oldestbo]=FillBOs(BOlist, tempEL, days, DI, WIP, Z)

%Check BO list, fill the oldest BO if one exists
if size(BOlist,1)>0 %backorders exist(ed) for some types of parts
    tempbo=[];
    for i=1:size(BOlist,1)
        if (BOlist(i,2)==tempEL(Z,1)) && (BOlist(i,5)==days+.000001) %part BO
hasn't been filled yet
            tempbo=[BOlist(i,:);tempbo]; %a list of all open BOs for the right
kind of part
        end
    end

    %Account for BO's already about to be filled
    %if a part is WIP at the location, or DI to the location, a BO will be
temporarily nullified (reduces shipping)
    if size(tempbo,1)>0
        tempbo=sortrows(tempbo,-4); %oldest BOs on top

        %if a part is DI or WIP, delete the oldest BO
        localstock=DI(tempEL(Z,1),:)+WIP(tempEL(Z,1),:); %row vector of WIP+DI
for a single part
        for u=1:size(localstock,2)  %for all locations
            deletenum=localstock(1,u); %count the number of parts en-route or
WIP

            for w=size(tempbo,1):-1:1  %going backwards up the list of
outstanding BOs, oldest to newest
                if size(tempbo,1)>0
                    if (tempbo(w,3)==u)&&(deletenum>0)
                        tempbo(w,:)=[];  %delete the row if the BO is about to
be filled by already DI or WIP on location
                        deletenum=deletenum-1;
                    end
                end
            end
        end
     end

    if size(tempbo,1)>0
        [C, m]=min(tempbo(:,4)); %C is the value, m is the index, finds
earliest startbo value
        oldestbo=tempbo(m,3); %returns the base having the oldest BO for that
part in the BOlist
    else
        oldestbo=0;
    end
else
    oldestbo=0;
end

end
```

flushout

```
function[stocklevels]=flushout(partnum, basenum, locnum, stocklevels, Pswitch,
DDR, RT, FromTo, NRTS, SEBOs)
OldSEBOs=SEBOs;
for k=1:partnum
    counter=0; %initialize to enter while loop
    while (counter<basenum) && (k<partnum+1)  %There are bases that benefit
from redistribution, and there are parts left to evaluate
        templevels=stocklevels;
        if templevels(k,1)>0  %there are parts to distribute
            List=zeros(1,basenum+1); %create row vector of SEBOs

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            for u=2:basenum+1  %for each of the bases, not the depot
                templevels(k,1)=templevels(k,1)-1;  %pull one from the depot
                templevels(k,u)=templevels(k,u)+1;  %put it at the base of the
moment

                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                %find system EBOs
                Pipe=pipeline(partnum, locnum, Pswitch, templevels, DDR, RT,
FromTo, NRTS);
                SEBOs=zeros(1,1);
                for j=2:locnum
                    SEBOs(1,1)= SEBOs(1,1)+(nbEBO(Pswitch, templevels(k,j),
Pipe(k,j)));
                end
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                List(1,u)=SEBOs;

                templevels(k,1)=templevels(k,1)+1;  %Reset
                templevels(k,u)=templevels(k,u)-1;
                counter=counter+1;
            end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            [V,I]=min(List(1,2:locnum));
            I=I+1;

            if List(1,I)<OldSEBOs(k,1)
                stocklevels(k,1)=stocklevels(k,1)-1;
                stocklevels(k,I)=stocklevels(k,I)+1;
                OldSEBOs(k,1)=List(1,I);
                counter=0;
            end
        else
            k=k+1;
        end
    end
end
flushout=stocklevels;
end
```

```
halfs
numtrial=size(timetrials,1);
    clear halves difference
    for  q=1:1
        ebosimBEBO;
        halves(q,1)=hlf
        difference(q,1)=(Xb-AnBEBOs(1,4))
    end
```

Holes

```
function[col]=Holes(permlevels,S0,laterals,DR,fixer,locnum)

fractions=[];
for i=1:locnum
    if permlevels(1,i)==0
        fractions(1,i)=1;  %if the location has not been assigned a level, it
shouldn't go there.
    else
        fractions(1,i)=S0(1,i)/permlevels(1,i); %the percent of the level
already accounted for at each location
    end
end

if DR==1  %part was fixed at depot and can go to any base
    [C,I]=min(fractions); %C is value, I is index

    if fractions(1,I)<1
        tempcol=I; %it goes where needed
    else
        tempcol=1;  %it stays at depot
    end

else %part was fixed at base, need to check laterals
    if laterals==0  %part can be kept at base or sent to depot
        if fractions(1,1)<fractions(1,fixer)
            tempcol=1; %send to depot
        else
            tempcol=fixer; %keep at base
        end

    else %part can go anywhere
        [C,I]=min(fractions); %C is value, I is index

        if fractions(1,I)==fractions(1,fixer)
            tempcol=fixer; %all things equal, keep it at the base
        else
            tempcol=I; %if there really is a min, send it there
        end
    end
end
col=tempcol;
end
```

LatRes

```
function[out]=LatRes(OH, DI, WIP, tempddr, FT, basenum, part, base)

if sum(OH(part,:))>0
    %Which bases have at least one unit OH?
    gotit=zeros(2,basenum);
    for i=1:basenum
        if OH(part,i+1)>0
            gotit(1,i)=1;
        end
    end

    %Which base can best release it?
    for i=1:basenum
        if gotit(1,i)==1

gotit(2,i)=(OH(part,i+1)+DI(part,i+1)+WIP(part,i+1))/tempddr(part,i+1); %could
other rules be used?
        end
    end
    gotit=[gotit;(1:basenum)];
    gotit=[gotit;random('unif',0,1,[1,basenum])]';
    gotit=sortrows(gotit,4);

    [C,I]=max(gotit(:,2));
    Tempbase=gotit(I,3)+1;

    %Is is better to wait on the depot?
    if WIP(part,1)>0
        if (tempddr(part,1)/2+FT(1,base))<FT(Tempbase,base)
            Tempbase=9999;
        end
    end

    out=Tempbase;
else
    out=9999;
end
end
```

Levelsa

```
%This code successfully reproduced Sherbrooke's Table 3-3 on pg. 52 using a
%single depot and 5 bases of equal ddrs for one part (using budget of 8).

%Assumes all demands originate at the bases and cascade up, and FromTo is same
for all parts

%Inputs:  DDR, NRTS, RT  (times in years, demands per year)
    %row=part, column=location
%Inputs:  Cost
    %row=part
%Inputs:  FromTo (time in years)
    %row=location, column=location
tic
clc;
clear partnum locnum stocklevels Mcirf Mdep basepart cirfpart depotpart Pipe;
warning off
warning1=0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Budget=8;%   <----------------------User Input
tempmoney=Budget;
basenum=size(DDR,2)-1;

Pswitch=1;%   <----------------------User Input , set to 1 for poisson values
instead of neg bin
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

partnum=size(DDR,1);
locnum=size(DDR,2);
d=ceil(sqrt(partnum));
close all;

firstbase=locnum-basenum+1;
lastbase=locnum;

%find avg demand on depot for each part, updates depot DDRs
DDR(:,1)=updatedepotddr(partnum, locnum, DDR, NRTS);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%AAM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%stocklev
els=zeros(partnum, 1);%does not have to be zeros, could be anything
storeEBOspo=zeros(partnum, 1);
storeEBOspz=zeros(partnum, 1);

[meanDDR, meanRT, meanFromTo, meanNRTS]= means(DDR, RT, FromTo, NRTS);

while Budget>0
    for m=1:partnum
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            %NOTE:  If the budget is too large, the sortvalues will eventually
become meaningless.
                    %When this happens, the budget is expended at random.

            %First pass calculates EBOs for the s+0 stocklevels at avg base

            templevels=stocklevels;

            Pipe=pipeline(partnum, 1, Pswitch, templevels, meanDDR, meanRT,
meanFromTo, meanNRTS);
```

```
            %Find total base EBOs for each part at s Plus Zero templevel
            storeEBOspz(m,1)=0;
                for i=1:partnum
                    storeEBOspz(m,1)= storeEBOspz(m,1)+(nbEBO(Pswitch,
templevels(i,1), Pipe(i,1)));
                end

            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %Second pass calculates EBOs for the s+1 stocklevels at avg base
            templevels=zeros(partnum,1);
            templevels(m,1)=1;
            templevels=templevels+stocklevels;

            Pipe=pipeline(partnum, 1, Pswitch, templevels, meanDDR, meanRT,
meanFromTo, meanNRTS);

            %Find EBOs for S Plus One templevel at avg base
            storeEBOspo(m,1)=0;
                for i=1:partnum
                    storeEBOspo(m,1)= storeEBOspo(m,1)+(nbEBO(Pswitch,
templevels(i,1), Pipe(i,1)));
                end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    end

    [Row, Col, flag]=sortvalues(storeEBOspz, storeEBOspo, partnum, 1, Cost);

    if flag==1
        warning1=1;
    end

    %Update Stock Levels until budget is expended
    stocklevels(Row,Col)=stocklevels(Row,Col)+1;
    Budget=Budget-Cost(Row);

    Pipe=pipeline(partnum, 1, Pswitch, stocklevels, meanDDR, meanRT,
meanFromTo, meanNRTS);

    %Report total System EBOs; doesn't mean much for avg base, interesting
    %SEBOs=zeros(partnum,1);
    %for i=1:partnum
    %    SEBOs(i,1)= SEBOs(i,1)+(nbEBO(Pswitch, stocklevels(i,1), Pipe(i,1)));
    %end
    %stocklevels
    %SEBOs
end
maxparts=stocklevels

clear meanDDR meanRT meanFromTo meanNRTS;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%RBL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
stocklevels=zeros(partnum, locnum);%does not have to be zeros, could be
anything
templevels=zeros(partnum, locnum);
storeEBOspo=zeros(partnum, locnum);
storeEBOspz=zeros(partnum, locnum);
```

```
for m=1:partnum
    while maxparts(m,1)>0
        for n=1:locnum

            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %First pass calculates EBOs for the s+0 stocklevels for a part
            templevels=stocklevels;

            Pipe=pipeline(partnum, locnum, Pswitch, templevels, DDR, RT,
FromTo, NRTS);

            %Find total base EBOs for each part's and each location's S Plus
Zero templevel
            storeEBOspz(m,n)=0;
            for j=2:locnum
                storeEBOspz(m,n)= storeEBOspz(m,n)+(nbEBO(Pswitch,
templevels(m,j),Pipe(m,j)));
            end

            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %Second pass calculates EBOs for the s+1 stocklevels for a part
            templevels=zeros(partnum,locnum);
            templevels(m,n)=1;
            templevels=templevels+stocklevels;

            Pipe=pipeline(partnum, locnum, Pswitch, templevels, DDR, RT,
FromTo, NRTS);

            %Find ttlbase EBOs for each location's S Plus One templevel for a
part
            storeEBOspo(m,n)=0;
            for j=2:locnum
                storeEBOspo(m,n)= storeEBOspo(m,n)+(nbEBO(Pswitch,
templevels(m,j),Pipe(m,j)));
            end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        end

        [Row, Col, flag]=sortvalues(storeEBOspz(m,:), storeEBOspo(m,:), 1,
locnum, Cost(m,1));

        if flag==1
            warning1=1;
        end

        %Update Stock Levels
        stocklevels(m,Col)=stocklevels(m,Col)+1;
        maxparts(m,1)= maxparts(m,1)-1;

        Pipe=pipeline(partnum, locnum, Pswitch, stocklevels, DDR, RT, FromTo,
NRTS);

        BEBOs=zeros(partnum,1);
        for i=1:partnum
            for j=2:locnum
                BEBOs(i,j)=(nbEBO(Pswitch, stocklevels(i,j), Pipe(i,j)));
            end
        end

        %Find System EBOs
```

```
        SEBOs=zeros(partnum,1);
        for i=1:partnum
            for j=2:locnum
                SEBOs(i,1)= SEBOs(i,1)+(nbEBO(Pswitch, stocklevels(i,j),
Pipe(i,j)));
            end
        end

        sebovect=SEBOs;

        %stocklevels
        %BEBOs
        %SEBOs
    end
end
permlevels=stocklevels;
stocklevels
%BEBOs
SEBOs

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%Need to account for flushout phenomenon because a marginal analysis was
used in leiu of exhaustive search
%This tends to be an issue with identical bases; happens infrequently, but just
in case
%see sherbrooke, pg 54.

fprintf('FLUSHOUT PROCEDURE\r')
stocklevels=flushout(partnum, basenum, locnum, stocklevels, Pswitch, DDR, RT,
FromTo, NRTS, SEBOs);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


fprintf('FINAL RESULT\r')

Pipe=pipeline(partnum, locnum, Pswitch, stocklevels, DDR, RT, FromTo, NRTS);

BEBOs=zeros(partnum,1);
for i=1:partnum
    for j=2:locnum
        BEBOs(i,j)=(nbEBO(Pswitch, stocklevels(i,j), Pipe(i,j)));
    end
end

%Find System EBOs
SEBOs=zeros(partnum,1);
for i=1:partnum
    for j=2:locnum
        SEBOs(i,1)= SEBOs(i,1)+(nbEBO(Pswitch, stocklevels(i,j), Pipe(i,j)));
    end
end

sebovect=SEBOs;
AnBEBOs=BEBOs;

stocklevels
BEBOs
SEBOs
TotalEBOs=sum(SEBOs)

 toc
```

```
if warning1==1
    fprintf('Some random assignment due to machine epsilon differences in sort
values; budget too large\r')
end
Budget=tempmoney;
%

% for i=1:partnum
%     subplot(d,d,i)
%     bar(stocklevels(i,:)','g')
%     xlabel('location');
%     ylabel(sprintf('Levels for part%i ',i));
% end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear partnum locnum Mcirf Mdep i j k bigsv maxparts counter templevels
OldSEBOs;
clear b c d C D storeEBOspo storeEBOspz prow Pipe tempmoney;
clear m n Row Col sortvalues templevels cirfEBOs depotEBOs ans flag warning1;
```

master

```
%Drives EBO research with and without lateral resupply
clc;
rownum=50;   %<----------------user input, number of mock parts
colnum=19;
neweventlist=1;

ddrlow=1;  %actually, this is annual demand
ddrhigh=52;

NRTSlow=0;
NRTShigh=1;

DRTlow=(15/365);
DRThigh=(90/365);

BRTlow=(3/365);
BRThigh=(45/365);

OSTdblow=(3/365);
OSTdbhigh=(45/365);

OSTbblow=(3/365);
OSTbbhigh=(45/365);

Costlow=.5;
Costhigh=2;

R=zeros(rownum,14);
T=[];
R=random('uniform',0,1,[rownum,colnum]);
Rcopy=R;

for i=1:rownum
    for j=1:5
        R(i,j)=R(i,j)*(ddrhigh-ddrlow)+ddrlow;
    end
end
for i=1:rownum
    for j=6:10
        R(i,j)=R(i,j)*(NRTShigh-NRTSlow)+NRTSlow;
    end
end

for i=1:rownum
    R(i,11)=R(i,11)*(DRThigh-DRTlow)+DRTlow;
end
for i=1:rownum
    for j=12:16
        R(i,j)=R(i,j)*(BRThigh-BRTlow)+BRTlow;
    end
end
for i=1:rownum
    R(i,17)=R(i,17)*(OSTdbhigh-OSTdblow)+OSTdblow;
    R(i,18)=R(i,18)*(OSTbbhigh-OSTbblow)+OSTbblow;
end
for i=1:rownum
    R(i,19)=R(i,19)*(Costhigh-Costlow)+Costlow;
end

z=zeros(rownum,1);
```

```
ddrfake=[z,R(:,1:5)];
NRTSfake=[z,R(:,6:10)];
a=updatedepotddr(rownum, 6, ddrfake, NRTSfake);

T(:,1)=a(:,1); %ddrdepot
clear a z ddrfake NRTSfake;

for i=1:rownum
    T(i,2)=mean(R(i,1:5)); %ddrbar
    T(i,3)=std(R(i,1:5)); %ddr_sigma
    T(i,4)=mean(R(i,6:10));%NRTSbar
    T(i,6)=std(R(i,6:10)); %NRTSsigma
    T(i,7)=mean(R(i,12:16)); %BRTbar
    T(i,9)=std(R(i,12:16)); %BRT_sigma
    T(i,12)=T(i,2)*T(i,4);
    T(i,13)=T(i,2)*T(i,7);
    T(i,14)=T(i,4)*T(i,7);
end

xd=rownum;
Rd=R;
clear rownum ddrlow ddrhigh NRTSlow NRTShigh DRTlow DRThigh BRTlow BRThigh
clear OSTdblow OSTdbhigh OSTbblow OSTbbhigh Costlow Costhigh i j R With Without
Response

for ii=1:xd
    tic
    DDR=[];
    NRTS=[];
    FromTo=[];
    RT=[];
    Cost=[];

    DDR=[T(ii,1),Rd(ii,1:5)];
    NRTS=[0,Rd(ii,6:10)];
    FromTo=ones(6,6)*Rd(ii,18);
    for jj=1:6
        FromTo(jj,1)=Rd(ii,17);
        FromTo(1,jj)=Rd(ii,17);
        FromTo(jj,jj)=0;
    end
    RT=Rd(ii,11:16);
    Cost=Rd(ii,19);

    %run bosim without lateral resupply
    laterals1=0;
    bosim;
    Without(ii,1)=SimulatedEBOs;

    %run bosim with lateral resupply
    laterals1=1;
    bosim;
    With(ii,1)=SimulatedEBOs;
    toc
    ii
end
A=[Rd,T];
Response=Without-With;
Response=[With,Without,Response];

clear ii jj xd;
```

means

```
function [meanDDR, meanRT, meanFromTo, meanNRTS]=means(DDR, RT, FromTo, NRTS)
locnum=size(DDR,2);
partnum=size(DDR,1);
baseDDR=DDR;
baseRT=RT;
baseFromTo=FromTo;
baseNRTS=NRTS;
    baseDDR(:,1)=[];
    baseRT(:,1)=[];
    baseNRTS(:,1)=[];
meanDDR=mean(baseDDR')';
meanRT=mean(baseRT')';  %could be demand-weighted
meanFromTo=mean(baseFromTo(2:locnum,1))*ones(partnum,1); %could be demand-
weighted
meanNRTS=mean(baseNRTS')'; %could be demand weighted
end
```

nbebo (adopted from VBA code used in HQ AFMC/A9A)

```
function[out]=nbEBO(Pswitch, stock, PipeQ)
% Calculate EBOs with Negative Binomial assumption
% PipeQ=Pipeline Quantity,  VTM = Variance to mean ratio,   stock=level of
stock

vtm=VarToMean(Pswitch, PipeQ);
Bo = PipeQ;
px = cdfNegBin(0, vtm, PipeQ);
for i = 1:stock
    Bo = Bo - 1 + px;
    px = cdfNegBin(i, vtm, PipeQ);
end
out = Bo;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[out]=cdfNegBin(stock, vtm, PipeQ)
% This function calculates a Negative Binomial cdf
% PipeQ=Pipeline Quantity;  VTM = Variance to mean ratio;   stock=level of
stock
% parameters f=r,g=p, and stock=v in standard negbin interpretation

%Parameters
    g = 1 / vtm;
    f = PipeQ / (vtm - 1);
    px = exp(f * (log(1) - log(vtm)));
    psum = 0;
  for i = 1:(stock + 1)
    psum = psum + px;
    px = exp(log(px) + log(f + i - 1) + log(1 - g) - log(i));
  end
  %prevents LN(0) in sort values
  if psum <= 0.000001
        psum = 0.000001;
  end
    out = psum;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[out]=VarToMean(Pswitch, PipeQ)
%Calculate variance to mean ratio from pipeline quantity PipeQ using Hill
Stevens method
    if Pswitch==1
        VarToMean=1.000001;
    else
        if PipeQ == 0
            VarToMean = 1.000001;
        else
            VarToMean = 1.132477 * ((PipeQ) ^ 0.3407513);
        end

        if VarToMean <= 1
            VarToMean = 1.000001;
        end

        if VarToMean > 5
            VarToMean = 5;
        end
    end
    out=VarToMean;
end
```

pipeline

```
function[out]=pipeline(partnum, locnum, Pswitch, levels, DDR, RT, FromTo, NRTS)

%Creates matrix of pipelines, rows are parts, columns are locations, the first
column is the depot's pipeline

%Compute pipelines:
            Pipe=zeros(partnum,locnum);
            for i=1:partnum
                depotEBOs=(nbEBO(Pswitch,
levels(i,1),DDR(i,1)*RT(i,1)))/DDR(i,1);
                if DDR(i,1)==0
                   depotEBOs=0;
                end
                for j=2:locnum
                    b=RT(i,j);
                    d=FromTo(1,j)+depotEBOs;
                    Pipe(i,j)=DDR(i,j)*((1-NRTS(i,j))*b+NRTS(i,j)*d); %base
pipelines
                end
                Pipe(i,1)=DDR(i,1)*RT(i,1); %depot pipeline
            end
out=Pipe;
end

%As stock is added pipelines decrease because the depot backorder component
%is reduced.
```

promeval

```
function[promote]=promeval(reps, counts, lamdat, upperbound)
%returns the replicaton demand size to be promoted one higher.
%we want to promote an "m-1" to an "m" meaning that a
%replication with "m-1" demands should get the next RANDOM demand to
%be allocated.  Subsequently the number of demands falling in each
%replication (randomly) will be Poisson.

    Pcdf=[];
    store=0;
    for i=1:upperbound+1
        Pcdf(i,2)=poisscdf(i-1,lamdat);   % for 0,1,2....upperbound ... the
second column is the Poisson CDF
        Pcdf(i,1)=counts(i,2)/reps+store; % for 0,1,2....upperbound ... the
first column is the empirical CDF
        store=Pcdf(i,1);
    end


    Deltas=[counts(:,1), abs(Pcdf(:,2)-Pcdf(:,1))];  %counts(:,1) is indices,
1:upperbound

    %Finds promotion options
    Deltatemp=Deltas;
    for i=upperbound+1:-1:1   %steps backwards through counts and Deltatemp
(they start as the same size)
        if counts(i,2)==0       %if no replications have 'i' demands, they
cannot be promoted into replications having 'i+1' demands
            Deltatemp(i,:)=[];
        end
    end  %if no replications had 7 demands, [7,0] would have been deleted
because [8,0] is infeasible; only the feasible set of promotion options remains

    [C, m]=max(Deltatemp(:,2)); %C is the maximum value (needed), m is the row
index (not needed)

    for i=1:upperbound+1
        if C==Deltas(i,2) %find where C occurs in Deltas
            m=i;  %m is the index where that happens
            break
        end
    end
    promote=m-1; %promote up to m
end
```

Regressit

```
%inputs are regressor matrix A, response vector Response, Vnames, Clust <------
----user input
%Clust is a list of what clusters each datapoint belongs to
%it doesnn't matter if A has leading ones, performs various regression
%functions

clc;
close all;
clear Bhat Yhat e SSres MSres SSreg MSreg SSt Fo Fstat alpha C H X r d;
clear ePRESS Si2 Rstud t nvector groupnum Ybarvector SSpe ANOVA Xhatp;
clear Yhatp U Z xi xerror yerror Tcrit BoxCoxusedlamda BoxCoxusedlog;
clear leveragepoints Cooks DFFITS Cooksinfluence DFFITSinfluence;
clear DFBETASinfluence DFBETAS DFBETAcountries V R Z Rstud ePRESS;
clear Yhata PRESS CIforBhat groupcity30 groupcity60 groupsqEuc30 groupsqEuc60;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Switches
    GRAPHS=1;% 0 is off<----------user input
    BOXCOX=1;% 0 is off<----------user input
    ALLREG=1;% 0 is off<----------user input
    LofFit=0;% 0 is off<----------user input
    Warnng=0;% 0 is off<----------user input
    GENLSQ=0;% 0 is off<----------user input
    wlsreg=0;% 0 is off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%add a column of ones to A if it needs one and get sizes of A (n by p)
    Y=Response;
    n=size(A,1);
    if A(:,1)~=ones(n,1)
        A=[ones(n,1) A];
    end
    p=size(A,2);
    globalp=p;
    Filter = int8(ones(1,p));



    %standardize all data to -1:1
    Amax=zeros(1,p);
    Amin=Amax;
    for i=2:p
        Amax(1,i)=max(A(:,i));
        Amin(1,i)=min(A(:,i));
    end

    for j=2:p
        for i=1:n
            Abba(i,j)=((A(i,j)-Amin(1,j))/(Amax(1,j)-Amin(1,j)))*2-1;
        end
    end
    A(:,2:p)=Abba(:,2:p);

    clear i j Abba Amax Amin

    %Filter out certain regressors - uncomment to "eliminate"
%     Filter(1,1)=0;% filter B0<----------user input
%     Filter(1,2)=0;% filter B1<----------user input
%     Filter(1,3)=0;% filter B2<----------user input
%     Filter(1,4)=0;% filter B3<----------user input
%     Filter(1,5)=0;% filter B4<----------user input
```

```
%     Filter(1,6)=0;% filter B5<----------user input
%     Filter(1,7)=0;% filter B6<----------user input
%     Filter(1,8)=0;% filter B7<----------user input
%     Filter(1,9)=0;% filter B8<----------user input
%     Filter(1,10)=0;% filter B9<----------user input
%     Filter(1,11)=0;% filter B10<----------user input
%     Filter(1,12)=0;% filter B11<----------user input
%     Filter(1,13)=0;% filter B12<----------user input
%     Filter(1,14)=0;% filter B13<----------user input
%     Filter(1,15)=0;% filter B14<----------user input
%     Filter(1,16)=0;% filter B15<----------user input
%     Filter(1,17)=0;% filter B16<----------user input
%     Filter(1,18)=0;% filter B17<----------user input
%     Filter(1,19)=0;% filter B18<----------user input
%     Filter(1,20)=0;% filter B19<----------user input
%     Filter(1,21)=0;% filter B20<----------user input
%     Filter(1,22)=0;% filter B21<----------user input
%     Filter(1,23)=0;% filter B22<----------user input
%     Filter(1,24)=0;% filter B23<----------user input
%     Filter(1,25)=0;% filter B24<----------user input
%     Filter(1,26)=0;% filter B25<----------user input
%     Filter(1,27)=0;% filter B26<----------user input
%     Filter(1,28)=0;% filter B27<----------user input
%     Filter(1,29)=0;% filter B28<----------user input
%     Filter(1,30)=0;% filter B29<----------user input
%     Filter(1,31)=0;% filter B30<----------user input



    X=A;
    for i=p:-1:1
        if Filter(1,i)==0
            X(:,i)=[];
        end
    end
    p=size(X,2);

    explist=ones(1,p);
    Xform=int8(zeros(1,p));
    %Pick regressors to transform - uncomment to Xform via Box-Tidwell
%%%%%%%%%%%%%Do not transform x0 via Box Tidwell
%         Xform(1,2)=1;% Xforms x1 via Box-Tidwell<----------user input
%         Xform(1,3)=1;% Xforms x2 via Box-Tidwell<----------user input
%         Xform(1,4)=1;% Xforms x3 via Box-Tidwell<----------user input
%         Xform(1,5)=1;% Xforms x4 via Box-Tidwell<----------user input
%         Xform(1,6)=1;% Xforms x5 via Box-Tidwell<----------user input
%         Xform(1,7)=1;% Xforms x6 via Box-Tidwell<----------user input
%         Xform(1,8)=1;% Xforms x7 via Box-Tidwell<----------user input
%         Xform(1,9)=1;% Xforms x8 via Box-Tidwell<----------user input
%         Xform(1,10)=1;% Xforms x9 via Box-Tidwell<----------user input
%         Xform(1,11)=1;% Xforms x10 via Box-Tidwell<----------user input
%         Xform(1,12)=1;% Xforms x11 via Box-Tidwell<----------user input
%         Xform(1,13)=1;% Xforms x12 via Box-Tidwell<----------user input
%         Xform(1,14)=1;% Xforms x13 via Box-Tidwell<----------user input
%         Xform(1,15)=1;% Xforms x14 via Box-Tidwell<----------user input
%         Xform(1,16)=1;% Xforms x15 via Box-Tidwell<----------user input
%         Xform(1,17)=1;% Xforms x16 via Box-Tidwell<----------user input
%         Xform(1,18)=1;% Xforms x17 via Box-Tidwell<----------user input
%         Xform(1,19)=1;% Xforms x18 via Box-Tidwell<----------user input
%         Xform(1,20)=1;% Xforms x19 via Box-Tidwell<----------user input
%         Xform(1,21)=1;% Xforms x20 via Box-Tidwell<----------user input
```

```
%          Xform(1,22)=1;% Xforms x21 via Box-Tidwell<----------user input
%          Xform(1,23)=1;% Xforms x22 via Box-Tidwell<----------user input
%          Xform(1,24)=1;% Xforms x23 via Box-Tidwell<----------user input
%          Xform(1,25)=1;% Xforms x24 via Box-Tidwell<----------user input
%          Xform(1,26)=1;% Xforms x25 via Box-Tidwell<----------user input
%          Xform(1,27)=1;% Xforms x26 via Box-Tidwell<----------user input
%          Xform(1,28)=1;% Xforms x27 via Box-Tidwell<----------user input
%          Xform(1,29)=1;% Xforms x28 via Box-Tidwell<----------user input
%          Xform(1,30)=1;% Xforms x29 via Box-Tidwell<----------user input
%          Xform(1,31)=1;% Xforms x30 via Box-Tidwell<----------user input


%variance stabilization
%if size(A,2)>2
%if A(1,3)==184
    %A(:,2)=A(:,2).^1.25; %x1
    %A(:,3)=A(:,3).^1.25; %x2
    %A(:,4)=A(:,4).^1.25; %x3
    %A(:,6)=A(:,6).^3; %x5
%end
%end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Warnng==0
    warning off;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Response vs regressor
%if GRAPHS==1
        k=1;
        r=1;
        jvector=zeros(p,1);
    d=ceil(sqrt(p));
    figure(21)
        for i=1:p-1
            subplot(d,d,i)

            while k<=globalp
                if Filter(i,k)==1
                    jvector(r,1)=k-1;
                    r=r+1;
                end
                k=k+1;

            end
            plot(A(:,i+1),Response,'ob', 'MarkerFaceColor','g')
%             xlabel(sprintf('x%i',jvector(i+1,1)));
%             axis([-1, 1, -.05, .15]);

%             %ylabel('Response');
%             title({sprintf('%s',char(Vnames(1,jvector(i+1,1))));' ';'Percent
EBO Reduction'});
        end
%end
clear k r jvector d

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%General Least Squares
if GENLSQ==1
    Save=X;
    V=cov(X');
    invV=(V)^-1;
```

```
    Bhatz=((X'*invV*X)^-1)*X'*invV*Y;
    K=(V)^.5;%  <--------- if covariances are negative, sqrts will be
imaginary.
    Bee=((K)^-1)*X;
    bigZ=Bee*Bhatz; % <-------------also imaginary

    SSresz=bigZ'*bigZ-Bhatz'*Bee'*bigZ;
    MSresz=SSresz/(n-p);

    SSregz=Bhatz'*Bee'*bigZ;
    MSregz=SSregz/(p-1);

    SStz=bigZ'*bigZ;

    %Calculate F statistic for model
    alpha=.90;
    Foz=MSregz/MSresz;
    Fstatz=finv(alpha,p-1,n-p);
    Fpvaluez=1-fcdf(Foz,p-1,n-p);

    %R-squared
    R2z=SSregz/SStz;
    R2adjz=1-(SSresz/(n-p))/(SStz/(n-1));

    %Build table (see pg 80 in book for explanation)
    glmANOVA=zeros(4,6);
    glmANOVA(1,1)=SSregz;  glmANOVA(1,2)=p-1;    glmANOVA(1,3)=MSregz;
glmANOVA(1,4)=Foz; glmANOVA(1,5)=Fpvaluez;
    glmANOVA(2,1)=SSresz;  glmANOVA(2,2)=n-p;    glmANOVA(2,3)=MSresz;
    glmANOVA(3,1)=SStz;    glmANOVA(3,2)=n-1;
    glmANOVA(4,1)=R2z;     glmANOVA(4,2)=R2adjz;

    clear  invV K Bee;
    X=Save;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%transformations on X -BoxTidwell
    alpha=.95;%                                    <----------user input
    y=Y;

    leading=ones(n,1);
    for i=1:p

        if Xform(1,i)==1
            x=[leading, X(:,i)];
            px=size(x,2);
            a=1;
            olda=10;

            while abs(olda-a)>.00005

                 %step 1
                bhat=((x'*x)\eye(px))*x'*y;
                yhat=x*bhat;
                C=(x'*x)\eye(px);
                SSres=y'*y-bhat'*x'*y;
                MSres=SSres/(n-px);
                To=abs(bhat(px,1)/sqrt(MSres*C(px,px)));
                Tcrit=tinv((alpha+(1-alpha)/2),n-px);
```

```
                %step 2
                w=x(:,px).*log(x(:,px));
                xw=[x,w];

                %step 3
                bhatw=((xw'*xw)\eye(px+1))*xw'*y;
                yhatw=xw*bhatw;

                %step 4
                Cx=(xw'*xw)\eye(px+1);
                SSresx=y'*y-bhatw'*xw'*y;
                MSresx=SSresx/(n-(px+1));

                Tox=abs(bhatw(px+1,1)/sqrt(MSresx*Cx(px+1,px+1)));
                Tcritx=tinv((alpha+(1-alpha)/2),n-(px+1));

                %step 5
                if To>Tcrit && Tox>Tcritx
                    a=bhatw(px+1,1)/bhat(px,1)+a;
                else
                    olda=a;
                end

                %step 6
                x(:,px)=x(:,px).^a;

            end
            explist(1,i)=a;
        end
    end

    for i=1:p
    explist(1,i)=round(explist(1,i)*2)/2;

        if explist(1,i)>2
            explist(1,i)=2;
        end
        if explist(1,i)<(-2)
            explist(1,i)=(-2);
        end
    end

    for i=1:p
        X(:,i)=X(:,i).^explist(1,i);
    end

clear x y olda To Tcrit Tox Tcritx w Cx bhatw;
clear MSresx SSresx MSres SSres yhatw bhat a xw yhat;
clear Xform leading %explist;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%transformations on Y -BoxCox

    if BOXCOX==1
        lamda=linspace(-2,2,21);
        lp=size(lamda,2);

        ydot=exp((1/n)*sum(log(Y)));

        for i=1:lp
            if lamda(1,i)~=0
                ytemp=(Y.^lamda(1,i)-1)./(lamda(1,i).*ydot^(lamda(1,i)-1));
```

```
            else
                ytemp=ydot.*log(Y);
            end
            bhat=((X'*X)\eye(p))*X'*ytemp;
            yhat=X*bhat;
            C=inv(X'*X);
            SSreslamda(1,i)=ytemp'*ytemp-bhat'*X'*ytemp;
        end

        lmin=min(SSreslamda);
        for i=1:lp
            if SSreslamda(1,i)==lmin
                location=i;
            end
        end
        if lmin~=0
            Y=(Y.^lamda(1,location)-1)/lamda(1,location);
            BoxCoxusedlamda=lamda(1,location)
        else
            Y=log(Y);
            BoxCoxusedlog=1
        end
        if GRAPHS==1
        figure(1)
        scatter(lamda,SSreslamda,'or', 'MarkerFaceColor','c');
        xlabel('Power Transformation Parameter Lamda');
        ylabel('SS_r_e_s'); title('SS_r_e_s vs. Lambda');
        end
    end
clear lp lmin ytemp location bhat yhat SSreslamda lamda ydot;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%fit model
    Bhat=((X'*X)\eye(p))*X'*Y;
    Yhat=X*Bhat;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%All possible regressions (p counts the intercept)
    if ALLREG==1
        clear All Nines Btemp mm nn U pall Bhata;

        AllReg=zeros(1,p);

        for i=1:p
            cmb=combntns(1:p,i);
            mm=size(cmb,1);
            nn=size(cmb,2);
            Btemp=zeros(mm,p);
            for j=1:mm
                for k=1:nn
                    Btemp(j,cmb(j,k))=1;
                end
            end
            AllReg=[AllReg;Btemp];
        end

        clear mm nn;
        mm=size(AllReg,1);
        nn=size(AllReg,2);

        U=X; %U holds the original X
        for i=1:mm
            for j=nn:-1:1
```

```
            if AllReg(i,j)==0
                X(:,j) = [];
            end
        end

        pall=size(X,2);
        Bhata=((X'*X)\eye(pall))*X'*Y;
        Yhata=X*Bhata;
         e=Y-Yhata;
         H=X*((X'*X)\eye(pall))*X';
            for s=1:n
                ePRESS(s,1)=(e(s,1)/(1-H(s,s)))^2;
            end

        All(i,1)=Bhata'*X'*Y -(Y'*ones(n,1))^2/n;              %SSreg
        All(i,2)=Y'*Y-Bhata'*X'*Y;                             %SSres
        All(i,3)=All(i,1)+All(i,2);                            %SSt
        All(i,4)=All(i,1)/All(i,3);                            %R2
        All(i,5)=1-(All(i,2)/(n-pall))/(All(i,3)/(n-1));       %R2adj
        All(i,6)=sum(ePRESS);                                  %PRESS

        X=U;
    end
    X=U;  %reset X

    numrgs=sum(AllReg')';
    tempM=ones(1,6);
    PandR2s=zeros(1,3);

    for i=1:p
        k=1;
        for j=1:mm
            if numrgs(j,1)==i
                tempM(k,:)=All(j,:);
                k=k+1;
            end
        end
            pickbiggest=max(tempM ,[] ,1);
            PandR2s(i,1)=i;                %the # of parameters used
            PandR2s(i,2)=pickbiggest(1,4); %R2
            PandR2s(i,3)=pickbiggest(1,5); %R2adj
    end

    if GRAPHS==1
    figure(2)
    plot(PandR2s(:,1),PandR2s(:,2),'r:o')
    hold on
    plot(PandR2s(:,1),PandR2s(:,3),'b:+')
    hold off
    xlabel('Number of Regression Coeficients');
    ylabel('R^2'); title('R^2 vs. Number of Regression Coefficients');
    legend('R^2','R^2 Adj.',2);
    end

    Nines=ones(mm,1)*9999999;
    All=[AllReg,Nines,All];
else
    clear All;
end
clear nn mm nopt i j k Bhata Nines U pall cmb AllReg Btemp numrgs tempM;
clear pickbiggest PandR2s;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%perform ANOVA
    alpha=.95;%                     <----------user input

    C=(X'*X)\eye(p);

    SSres=Y'*Y-Bhat'*X'*Y;
    MSres=SSres/(n-p);

    SSreg=Bhat'*X'*Y-(Y'*ones(n,1))^2/n;
    MSreg=SSreg/(p-1);

    SSt=SSreg+SSres;

    %Calculate F statistic for model

    Fo=MSreg/MSres;
    Fstat=finv(alpha,p-1,n-p);
    Fpvalue=1-fcdf(Fo,p-1,n-p);

    %Perform marginal T test for each Bhat
    for i=1:p
        To(i,1)=Bhat(i,1)/sqrt(MSres*C(i,i));
        StdErr(i,1)=sqrt(MSres*C(i,i));
        Tcrit(i,1)=tinv((alpha+(1-alpha)/2),n-p);
        Tpvalue(i,1)=2*(1-tcdf(abs(To(i,1)),n-p));
    end

    %R-squared
    R2=SSreg/SSt;
    R2adj=1-(SSres/(n-p))/(SSt/(n-1));

    %Multicollinearity
    Z=X;
    Z(:,1)=[];

    invR=corr(Z)\eye(p-1);
    VIF=zeros(p,1);
    for i=1:p-1
        VIF(i+1,1)= invR(i,i);
    end


    for i=1:p
        CIforBhat(i,1)=Bhat(i,1)-tinv((alpha+(1-alpha)/2),n-
p)*sqrt(MSres*C(i,i));
        CIforBhat(i,2)=Bhat(i,1);
        CIforBhat(i,3)=Bhat(i,1)+tinv((alpha+(1-alpha)/2),n-
p)*sqrt(MSres*C(i,i));
    end

    %Build table (see pg 80 in book for explanation)
    ANOVA=zeros(5+p,6);
    ANOVA(1,1)=SSreg;  ANOVA(1,2)=p-1;    ANOVA(1,3)=MSreg;  ANOVA(1,4)=Fo;
ANOVA(1,5)=Fpvalue;
    ANOVA(2,1)=SSres;  ANOVA(2,2)=n-p;    ANOVA(2,3)=MSres;
    ANOVA(3,1)=SSt;    ANOVA(3,2)=n-1;
    ANOVA(4,1)=R2;     ANOVA(4,2)=R2adj;
    for i=1:p
        ANOVA(5+i,1)=Bhat(i,1);
        ANOVA(5+i,2)=StdErr(i,1);
```

```
        ANOVA(5+i,3)=To(i,1);
        ANOVA(5+i,4)=Tcrit(i,1);
        ANOVA(5+i,5)=Tpvalue(i,1);
        ANOVA(5+i,6)=VIF(i,1);
    end

    %Distance Matrix
    for i=1:(size(A,1))
        for j=1:(size(A,1))
            for k=1:size(Bhat,1)
                d(k,1)=((Bhat(k,1)*(A(i,k)-A(j,k)))/sqrt(MSres))^2;
            end
            D(i,j)=sum(d);
        end
    end
    clear i j d;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Residuals
    e=Y-Yhat;
    H=X*((X'*X)\eye(p))*X';

    %Standardized residuals
    d=e/(MSres)^.5;

    %Studentized residuals
    for i=1:n
        r(i,1)=e(i,1)/(MSres*(1-H(i,i)))^.5;
    end

    %R-Student residuals
    for i=1:n
        Si2(i,1)=(((n-p)*MSres)-(((e(i,1))^2)/(1-H(i,i))))/(n-p-1);
    end
    for i=1:n
        Rstud(i,1)=e(i,1)/((Si2(i,1)*(1-H(i,i)))^.5);
    end

    %Residual Plots - check for normal, constant V, outliers
    if GRAPHS==1
        figure(3)
        subplot(2,2,1)
        scatter(Yhat,e); xlabel('Fitted values');
        ylabel('Residuals'); title('Residuals vs. Fits');
        subplot(2,2,2)
        scatter(Yhat,d); xlabel('Fitted values');
        ylabel('Standardized Residuals'); title('Standardized Residuals vs.
Fits');
        subplot(2,2,3)
        scatter(Yhat,r); xlabel('Fitted values');
        ylabel('Studentized Residuals'); title('Studentized Residuals vs.
Fits');
        subplot(2,2,4)
        scatter(Yhat,Rstud); xlabel('Fitted values');
        ylabel('R-Student Residuals'); title('R-Student Residuals vs. Fits');

        figure(4)
        residuals=normplot(e); xlabel ('Residuals');
        %     subplot(2,2,2)
        %     normplot(d); xlabel('Standardized Residuals');
        %     ylabel('Probability'); title('Normal Probability Plot for
Standardized Residuals');
```

```
        %     subplot(2,2,3)
        %     normplot(r); xlabel('Studentized Residuals');
        %     ylabel('Probability'); title('Normal Probability Plot for
Studentized Residuals');
        %     subplot(2,2,4)
        %     normplot(Rstud); xlabel('R-Student Residuals');
        %     ylabel('Probability'); title('Normal Probability Plot for R-
Student Residuals');
    end

    %Partial Residual Plots
    clear estar s;

    if GRAPHS==1
        for i=2:p
            estar(:,i)=e+Bhat(i,1)*X(:,i);
        end

        k=1;
        s=1;
        jvector=zeros(p,1);
        d=ceil(sqrt(p-1));
        figure(5)
            for i=1:p-1
                subplot(d,d,i)

                while k<=globalp
                    if Filter(i,k)==1
                        jvector(s,1)=k-1;
                        s=s+1;
                    end
                        k=k+1;
                end

             plot(X(:,i+1),estar(:,i+1),'hr', 'MarkerFaceColor','k')
             xlabel(sprintf('x%i',jvector(i+1,1)));
             ylabel('Partial Residual');
             title({'Partial Residual Plot for';
sprintf('%s',char(Vnames(1,jvector(i+1,1))))})

            end
            clear estar s;
    end

    %Cooks Distance
    k=1;
    for i=1:n
    Cooks(i,1)=r(i,1)^2*H(i,i)/(p*(1-H(i,i)));
        if Cooks(i,1)>1
            Cooksinfluence(k,1)=i;
            Cooksinfluence(k,2)=Cooks(i,1);
            Cooksinfluence(k,3)=r(i,1);
            k=k+1;
        end
    end

    %DFFITS
    k=1;
    for i=1:n
    DFFITS(i,1)=Rstud(i,1)*sqrt(H(i,i)/(1-H(i,i)));
        if abs(DFFITS(i,1))>2*sqrt(p/n)
```

```
            DFFITSinfluence(k,1)=i;
            DFFITSinfluence(k,2)=DFFITS(i,1);
            DFFITSinfluence(k,4)=Rstud(i,1);
            k=k+1;
        end
    end

    %DFBETAS
    R=C*X';
    d=size(X);
    limit=2/sqrt(n);
    k=1;
        for j=1:n
            for i=1:p
                CVX1=R(i,j);
                CVX2=R(i,:)*R(i,:)';
                CVX2=sqrt(CVX2);
                DFBETAS(j,i)=CVX1/CVX2*(Rstud(j,1))/sqrt(1-H(j,j));

                if abs(DFBETAS(j,i))>limit
                    DFBETASinfluence(k,1)=j;
                    DFBETASinfluence(k,i+1)=DFBETAS(j,i);
                    k=k+1;
                end
            end
        end

        T=DFBETASinfluence;
        u=size(T,1);
        v=size(T,2);

        for i=u:-1:2
            if T(i,1)==T(i-1,1)
                remember=T(i,1);
                T(i-1,:)=T(i-1,:)+T(i,:);
                T(i,:)=[];
                T(i-1,1)=remember;
            end
        end
        u=size(T,1);
        v=size(T,2);
        for i=1:u
            T(i,v+2)=Rstud(T(i,1),1);
        end
        DFBETASinfluence=T;

        clear u v remember T limit DFBETAS CVX1 CVX2 R;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Regressors
    %Fit models without the ith regressor to get errors
    for i=1:p
        U=X;
        xi=U(:,i);
        U(:,i)=[];
        I=((U'*U)\eye(p-1));
        Yhatp=U*((I)*U'*Y);
        yerror(:,i)=Y-Yhatp;
        Xhatp=U*((I)*U'*xi);
        xerror(:,i)=xi-Xhatp;
    end
```

```
    %Eliminate the Bo column from the error matrices
    yerror(:,1)=[];
    xerror(:,1)=[];

    %Make partial regression plots
    if GRAPHS==1
        k=1;
        r=1;
        jvector=zeros(p,1);
    d=ceil(sqrt(p));
    figure(6)
        for i=1:p-1
            subplot(d,d,i)

            while k<=globalp
                if Filter(i,k)==1
                    jvector(r,1)=k-1;
                    r=r+1;
                end
                k=k+1;
            end
            plot(xerror(:,i),yerror(:,i),'dr', 'MarkerFaceColor','g')
            xlabel(sprintf('Residuals for x%i',jvector(i+1,1)));
            ylabel('Response Residuals');
            title({'Partial Regression Plot for';
sprintf('%s',char(Vnames(1,jvector(i+1,1))))})

        end

        subplot(d,d,p)
        residuals=normplot(e); xlabel ('Residuals');
    end

    clear Xhatp Yhatp U Z xi xerror yerror I i j k r ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Weighted Least Squares
if wlsreg==1
clear a b;
    a=size(Clust,1);
    b=size(Clust,2);
    d=ceil(sqrt(b));

    for xyz=1:b

        Xbar=[];
        Vary=[];
        W=[];
        clear Betahatq yqhat Xbarq Varyq rightcol temp;

            numclust=max(Clust(:,xyz));
            temp=[];
            kk=1;
            ee=1;

            for j=1:numclust
                for t=1:a
                    if Clust(t,xyz)==j
                        temp(kk,:)=[X(t,:) Y(t,:)];
                        kk=kk+1;
                    end
                end
```

```
Xbar(ee,:)=[size(temp,1),ones(size(temp,1),1)'*temp/size(temp,1)];
                Vary(ee,:)=var(temp(:,size(temp,2)));

                ee=ee+1;
                temp=[];
                kk=1;
            end
                rightcol=size(Xbar,2);
                Xbar(:,rightcol)=[];
            %this far gets me xbars and var(y) for all size clusters 1 and up

            Xbarq=Xbar;
            Varyq=Vary;

            for j=numclust:-1:1
                if Varyq(j,1)==0
                    Xbarq(j,:)=[];
                    Varyq(j,:)=[];
                end
            end
            Xbarq(:,1)=[];

        Betaqhat=inv(Xbarq'*Xbarq)*Xbarq'*Varyq;
        yqhat=X*Betaqhat;

        W=(1./(X*Betaqhat));

        Bhatwls=inv(X'*diag(W)*X)*X'*diag(W)*Y;

        Yhatwls=X*Bhatwls;

        ewls=Y-Yhatwls;

        vert=sqrt(W).*ewls;
        horiz=sqrt(W).*Yhatwls;

        subplot(d,d,xyz)
        scatter(horiz,vert); xlabel('Weighted fits');
        ylabel('Weighted Residuals'); title(sprintf('Weighted Residuals for
Cluster Method %i',xyz));

        clear kk temp rightcol;
    end
        subplot(d,d,xyz+1)
        residuals=normplot(ewls); xlabel ('Weighted Residuals for Method 6');

    W=diag(W);
    V=inv(W);
    K=(V)^.5;
    bigw=(K^-1)*Y;
    UU=(K^-1)*X;
    SSresw=bigw'*bigw-Bhatwls'*UU'*bigw;
    MSresw=SSresw/(n-p);

    SSregw=Bhatwls'*UU'*bigw;
    MSregw=SSregw/(p-1);

    SStw=bigw'*bigw;
    zhat = UU*Bhatwls;
```

```
    %Calculate F statistic for model
     p=size(X,2);
     n=size(X,1);
     alpha=.90;
     Fow=MSregw/MSresw;
     Fstatw=finv(alpha,p-1,n-p);
    %Fpvaluew=1-fcdf(Fow,p-1,n-p);

    %Perform marginal T test for each Bhat
    alpha=.95;
    for i=1:p
        Tow(i,1)=Bhatwls(i,1)/sqrt(MSresw*C(i,i));
        StdErrw(i,1)=sqrt(MSresw*C(i,i));
        Tcritw(i,1)=tinv((alpha+(1-alpha)/2),n-p);
        Tpvaluew(i,1)=2*(1-tcdf(abs(To(i,1)),n-p));
    end

    Z=X;
    Z(:,1)=[];

    invR=corr(Z)\eye(p-1);
    VIFw=zeros(p,1);
    for i=1:p-1
        VIFw(i+1,1)= invR(i,i);
    end

    %R-squared
    R2w=SSregw/SStw;
    R2adjw=1-(SSresw/(n-p))/(SStw/(n-1));

    %Build table
    wlsANOVA=zeros(5,6);
    wlsANOVA(1,1)=SSregw;  wlsANOVA(1,2)=p-1;    wlsANOVA(1,3)=MSregw;
wlsANOVA(1,4)=Fow; %wlsANOVA(1,5)=Fpvaluew;
    wlsANOVA(2,1)=SSresw;  wlsANOVA(2,2)=n-p;    wlsANOVA(2,3)=MSresw;
    wlsANOVA(3,1)=SStw;    wlsANOVA(3,2)=n-1;
    wlsANOVA(4,1)=R2w;     wlsANOVA(4,2)=R2adjw;
    for i=1:p
        wlsANOVA(5+i,1)=Bhatwls(i,1);
        wlsANOVA(5+i,2)=StdErrw(i,1);
        wlsANOVA(5+i,3)=Tow(i,1);
        wlsANOVA(5+i,4)=Tcritw(i,1);
        wlsANOVA(5+i,5)=Tpvaluew(i,1);
        wlsANOVA(5+i,6)=VIFw(i,1);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Lack of Fit - need at least one replicate for this to run; it checks
    if LofFit==1
        groupnum=zeros(n,2);

        %find groups
%True Replicates
%         for i=1:n
%             if groupnum(i,1)~=999
%                 nvector(i,1)=1;
%                 groupnum(i,1)=999;
%                 groupnum(i,2)=i;
%                 for j=1:n
%                     if j~=i
```

```
%                         if X(j,:)==X(i,:)
%                             if groupnum(j,1)~=999
%                                 nvector(i,1)=nvector(i,1)+1;
%                                 groupnum(j,1)=999;
%                                 groupnum(j,2)=i;
%                             end
%                         end
%                     end
%                 end
%             end
%         end


%Clustered Replicates
groupnum=[999*ones(n,1),Clust(:,6)];  %takes the sixth column of the cluster
assignment matrix
nvector=zeros(max(Clust(:,6)),1);
for i=1:n
    for j=1:n
        if Clust(j,6)==i
            nvector(i,1)=nvector(i,1)+1;
        end
    end
end
        m=size(nvector,1);
        if m<n
            dfssres=n-p;
            dfsspe=n-m;
            dfsslof=m-p;

            %ybarvector
            ttlvector=zeros(m,2);
            for i=1:n

ttlvector(groupnum(i,2),1)=ttlvector(groupnum(i,2),1)+bigw(i,1);
                ttlvector(groupnum(i,2),2)=ttlvector(groupnum(i,2),2)+1;
            end
            Ybarvector=ttlvector(:,1)./ttlvector(:,2);

            %SSpe
            groupnum=[groupnum, Y];
            for i=1:n
                groupnum(i,4)=Ybarvector(groupnum(i,2),1);
            end

            SSpe=0;
            for i=1:n
            SSpe=SSpe+(groupnum(i,3)-groupnum(i,4))^2;
            end

            SSlof=SSresw-SSpe; %<---------------------adjusted for WLS
            lofFo= (SSlof/m-p)/(SSpe/n-m);
            lofFstat=finv(alpha,m-p,n-m);
            lofFpvalue = 1-fcdf(Fo,m-p,n-m);

            lofANOVA(1,1)=SSlof;  lofANOVA(1,2)=dfsslof;
lofANOVA(1,3)=SSlof/dfsslof;  lofANOVA(1,4)=Fo;  lofANOVA(1,5)=lofFpvalue;
            lofANOVA(2,1)=SSpe;   lofANOVA(2,2)=dfsspe;
lofANOVA(2,3)=SSpe/dfsspe;
        end
    end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear n p Filter Si2 SSres MSres SSreg MSreg SSt Fo Fstat ePRESS i r d t
wlsreg;
clear alpha disp residuals H Fpvalue C R2 R2adj dfssres dfsspe dfsslof;
clear m j N counter  lofFo e;
clear lofFpvalue SSlof SSpe StdErr To Tstat Tpvalue Bhat I VIF Rstud;
clear invR Tcrit X LofFit ALLREG BOXCOX GRAPHS globalp Warnng jvector;
clear DFFITS Cooks GENLSQ Foz Fpvaluez SStz SSresz SSregz MSresz MSregz;
clear Yhata Bhata Fstatz R2z R2adjz Save s clstrs xyz turnon numclust yqhat;
clear ee a b UU Z Tow Tcritw Tpavaluew VIFw StdErrw R2w R2adjw MSregw MSresw;
clear Tpvaluew SSregw SSresw SStw n p Fow Fpvaluew invR K V ewls h horiz vert
silh;
clear ttlvector Xbarq Vary Varyq Xbar zhat bigw Fstatw;
```

simlevel

```
clc;
Levelsa
%Budget must be updated in Levelsa      <--------------user input
%drives simulated readiness based leveling
clear g days

Funding=Budget;
fprintf('BEGINNING SIMULATED LEVELING\r')
partnum=size(DDR,1);
locnum=size(DDR,2);

stocklevels=zeros(partnum,locnum);
EBOmatrix=zeros(partnum,locnum);

while Funding>0
    tic

    %Create Levels:  What part?  What place?
    neweventlist=1;  %at least one eventlist per level allocation
    bosim;  %provides the Simulated EBOs needed for starter on each iteration
    starter=SimulatedEBOs*ones(partnum,locnum);  %each element is the same, the
number of EBOs with no additional stock
    for g=1:partnum
        for h=1:locnum
            stocklevels(g,h)=stocklevels(g,h)+1;
            bosim;
            EBOmatrix(g,h)=SimulatedEBOs;  %checks how adding one unit of each
type of part to each location changes system EBOs
            stocklevels(g,h)=stocklevels(g,h)-1;
        end
    end

    sv=(starter-EBOmatrix);
    for g=1:partnum
        sv(g,:)=sv(g,:)/Cost(g,1);
    end

    k=max(max(sv));
    for g=1:partnum
        for h=1:locnum
            if sv(g,h)==k
                Rowtemp=g;
                Coltemp=h;
            end
        end
    end

    stocklevels(Rowtemp,Coltemp)=stocklevels(Rowtemp,Coltemp)+1
    EBOmatrix(g,h)
    Funding=Funding-Cost(Rowtemp,1)
    toc
end
bosim;
EBOs=SimulatedEBOs

clear g h k sv Rowtemp Coltemp
clear locnum partnum starter Funding
beep
```

sortvalues

```
function[Row, Col, flag]=sortvalues(storeEBOspz, storeEBOspo, partnum, locnum,
Cost)

    flag=0;

    %Build Sort Value Matrix
    sortvalues=(storeEBOspz-storeEBOspo);
    for i=1:partnum
        for j=1:locnum
            sortvalues(i,j)=sortvalues(i,j)/Cost(i,1);
        end
    end

    %Largest element of sortvalues is found at (Row,Col)
    bigsv=max(max(sortvalues));
    for i=1:partnum
        for j=1:locnum
            if sortvalues(i,j)==bigsv
                Rowtemp=i;
                Coltemp=j;
            end
        end
    end
    if sortvalues(Rowtemp,Coltemp)<.000000000001  %if a tie exists, expend the
budget at random and put it at the depot
        Row=ceil(partnum*(random('unif',0,1,[1,1])));
        Col=1;
        flag=1;
    else
        Row=Rowtemp;
        Col=Coltemp;
        flag=0;
    end

    %in case of ties, randomly assign with preference to the depot
    if partnum==1
        tempsv=sortvalues;
        tempsv(2,:)=rand(1,locnum);
        tempsv(3,:)=1:locnum;
        tempsv=tempsv';
        for j=locnum:-1:1
            if tempsv(j,1)~=bigsv
                tempsv(j,:)=[];
            end
        end

        for j=1:size(tempsv,1)
            if tempsv(j,3)==1
                Row=Rowtemp;
                Col=1;
                flag=0;
                return
            end
        end
        tempsv=sortrows(tempsv,2);
        Row=Rowtemp;
        Col=tempsv(1,3);
        flag=0;
    end
end
```

updatedepotddr

```
function[out]=updatedepotddr(partnum, locnum, DDR, NRTS)

Mdep=zeros(partnum,1);
    for i=1:partnum
        for j=2:locnum
            Mdep(i,1)=Mdep(i,1)+DDR(i,j)*NRTS(i,j);
        end
    end
out=Mdep(:,1);
end
```

List of Symbols, Abbreviations and Acronyms

0	a context dependent index used to denote LRU or depot
A	aircraft availability
AAM	Aircraft Availability Model
B_i	backorder for a particular type of LRU
BEBO	base expected backorder
BO	backorder
BRT	base repair time
c	cost
CIRF	centralized intermediate repair facility
$D^{\#}$	number of non-operational planes
D_c	number of aircraft being cannibalized
D_{nc}	planes disabled that cannot be fixed through cannibalization
DDT	depot delay time
DEM	Distribution Enforcement Method
DI	due-in inventory
DoD	Department of Defense
Dof	degrees of freedom
DRIVE	Distribution and Repair in Variable Environments
E()	expectation operator

f	fraction of SRU demand
EBO	expected backorder
GAO	Government Accountability Office
g(t)	resupply time probability distribution
i	part type index
j	location number index
k	the most aircraft that can be down if cannibalization is performed to maximum extent
λ	mean demand rate
l	number of specific part types used per weapon system
LRU	line replaceable unit
n	number of types of LRUs on a weapon system
N	fleet size
NMCS	not mission capable supply
NRTS	not reparable this station
μ	pipeline quantity
m	average annual demand
METRIC	Multi-Echelon Technique for Recoverable Item Control
MS	mean square
O	average order and ship time
OH	on-hand inventory
OST	order and ship time
q_i	number of LRUs of a particular type installed on a weapon system

q_{ij}	conditional probability that a SRU *i* is responsible for failure of a LRU at base *j*
QPA	quantity per application
r	RTS rate
RBL	Readiness-Based Levels
RBS	Readiness-Based Sparing
RTS	reparable this station
s	stock
$\underline{s}$	a vector of stock levels for a type of plane
SEBO	system expected backorder
SRU	shop replaceable unit
SS	sum of squares
sv	sort value
τ	the mean of the resupply distribution
t	mean repair time
USAF	United States Air Force
V()	variance operator
VBO	variance of backorders
X	random variable for pipeline quantity
Z	number of parts in weapon system

Bibliography

Abell, John B., Grace M. Carter, Karen E. Isaacson, and Thomas F. Lippiatt. *Estimating Requirements for Aircraft Recoverable Spares and Depot Repair.* RAND, 1993.

Abreu, Roberto Carlos Borges. *The Effects of Variability in Demand and Time Parameters for Multi-Item, Multi-Echelon, Multi-Indenture Reparable Inventory Systems*, AFIT, WPAFB OH, 2002.

Banks, Jerry, John S. Carson II, Barry L. Nelson and David M. Nicol. *Discrete Event Simulation.* 4th edition, Pearson Prentice Hall, 2005.

Chakravorty, Satya Sunder. *A Simulation Study of the Performance of a Multi-Echelon Production-Distribution System with Feedback and Feedforward Controls*. University of Georgia, Athens GA, 1992.

Cochran, J. K. *Computing Small-Fleet Aircraft Availabilities Including Redundancy and Spares*. Computers and Operations Research, v 29, n 5, April 2002, p529-540.

Cohen, Morris A., Paul R. Kleindorfer, Hau Leung Lee. *Optimal Stocking Policies for Low Usage Items in Multi-Echelon Inventory Systems*. Naval Research Logistics Quarterly, v 33, n 1, February 1986, p17-38.

Defense Inventory, Opportunities Exist to Save Billions by Reducing Air Force's Unneeded Spare Parts Inventory. GAO-07-232. www.gao.gov/new.items/d07232.pdf 2007.

Evers, Philip T. *Heuristics for Assessing Emergency Transshipments*. European Journal of Operations Research, v 129, n 2, March 2001, p311-316.

Fisher, Warren W. *An Improved Simulation Model for Cannibalization Policy Performance Comparisons in a Complex Maintenance System*. Simulation, v 52, n 4, April 1989, p154-164.

Gaver, Donald P., Karen E. Isaacson and John B. Abell. *Estimating Aircraft Recoverable Spares Requirements with Cannibalization of Designated Items*. RAND, 1993.

Irani, Shahrukh. Personal Communications. The Ohio State University, Columbus OH 2004.

Lee, Hau L. *A Multi-Echelon Inventory Model for Repairable Items with Emergency Lateral Transshipments*. Management Science, v 33, n 10, October 1987, p1302-1316.

Long, Steve, Mark Abramson, Wayne Faulkner, Doug Blazer and Buddy Berry. *Readiness Based Leveling (RBL) Implementation Issues*. Air Force Logistics Management Agency Final Report LS96004000, 1996.

Miller, Louis W. and John B. Abell. *DRIVE (Distribution and Repair in Variable Environments)*. RAND, 1992.

Miller, Louis W. and John B. Abell. *Evaluations of Alternative Approaches to Central Stock Leveling*. RAND, 1995.

Muckstadt, John A. *Analysis and Algorithms for Service Parts Supply Chains*. Kluwer Springer Publishers, 2005.

O'Malley, T. J. *The Aircraft Availability Model: Conceptual Framework and Mathematics*. Logistics Management Institute Report AF201. June 1983.

Orsburn, Douglas K. *Spares Management Handbook*. McGraw-Hill, 1991.

Reynolds, Steve, Steve Long, Mark Abramson, Doug Blazer, Karl Kruse, and Wayne Faulkner. *Setting Recoverable Item Stock Levels*. Air Force Logistics Management Agency Final Report LS9500500, 1995.

Sherbrooke, Craig C. *Optimal Inventory Modeling of Systems*. Kluwer Academic Publishers, 2004.

Slay, F. Michael. *Artificial Retrospection, The Distribution Enforcement Method*. Logistics Management Institute, IR806R1, July 2007.

Stevens, R.J. and J.M. Hill. "*Estimating the Variance-to-Mean Ratio for Recoverable Items in the ALS Marginal Analysis Algorithms*." Working Paper #49, Systems Studies Branch, Office of DCS/Comptroller, HQ AFLC, WPAFB, 1973.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
03-03-2008	**Master's Thesis**	Sep 2007 – Mar 2008

4. TITLE AND SUBTITLE

SIMULATED MULTI-ECHELON READINESS-BASED INVENTORY LEVELING WITH LATERAL RESUPPLY

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Burnworth, Todd C.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 P Street, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOR/ENS/08M-23

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

HQ AFMC/A9A
Attn: Mr. Mike Niklas
4375 Chidlaw Rd. DSN: 787-7408
WPAFB OH 45433-7765 e-mail: mike.niklas@wpafb.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

For the past fifty years, U.S. Air Force reparable inventory has been allocated based on an analytic model developed by Dr. Craig C. Sherbrooke. Although versions of his model can be implemented easily with the help of a computer, the analytic approach fundamentally lacks the flexibility to address numerous logistics issues. This body of research will offer a novel alternative approach that will enable researchers to investigate currently unsolved logistics problems such as quantifying the benefits of lateral resupply.

15. SUBJECT TERMS

Inventory, Readiness-Based Levels, RBL, Readiness-Based Sparing, RBS, discrete event simulation, METRIC, reparables, multi-echelon, lateral resupply

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. John O. Miller (ENS)
U	U	U	UU	140	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565 x4326

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18